Video Foreground Segmentation with Deep Learning

Dimitrios Sakkos

Department of Engineering and Environment

Northumbria University

This dissertation is submitted for the degree of

Doctor of Philosophy

May 2020

I dedicate this thesis to my family,

ΓΙΩΡΓΟΣ ΒΑΣΙΛΟΠΟΥΛΟΣ

ΑΝΑΣΤΑΣΙΑ ΒΑΣΙΛΟΠΟΥΛΟΥ

ΝΙΚΗ ΒΑΣΙΛΟΠΟΥΛΟΥ

ΚΩΝΣΤΑΝΤΙΝΟΣ ΣΑΚΚΟΣ

ΧΡΥΣΟΥΛΑ ΣΑΚΚΟΥ

Declaration

I hereby declare that this thesis is the result of my own work and has not been submitted in any form for another degree or diploma at any university or other institution. I also confirm that this work fully acknowledges opinions, ideas and contributions from the work of others. Any ethical clearance for the research presented in this thesis has been approved. I declare that the Word Count of this Thesis is 21,207 words.

Dimitrios Sakkos

May 2020

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my principal supervisor Dr. Edmond Ho for supporting me in every way possible throughout my studies. I thank him for his invaluable advice over our meetings, for his endless support and for always taking the extra step to help me.

I would also like to thank Dr. Graham Sexton for his continuous support, guidance and valuable help. Furthermore, I thank Dr. Jungong Han for supporting me in a difficult crossroad of my PhD and for his valuable comments and suggestions.

I also feel the need to thank Prof. Ling Shao for introducing me to the area of Deep Learning in my first steps as a researcher and for his valuable guidance.

My thanks also go to Dr. Hubert Shum for always offering his insightful advice, from which I learned a lot. I also thank Dr. Paul Vickers, Dr. Allan Osborne and Dr. Nauman Aslam for supporting me when I needed them the most. Finally I would like to thank my former and current fellow PhD students from the University of Northumbria for their encouragement during my studies and the constructive discussions we have had. Especially, Daniel Organisciak, Kateřina Jandová and Kevin McCay for their tremendous support and encouragement.

Finally, I would have never been able to achieve this milestone without the never-ending love and support of my siblings, parents and grandparents, to whom I owe everything.

Abstract

This thesis tackles the problem of foreground segmentation in videos, even under extremely challenging conditions. This task comes with a plethora of hurdles, as the model needs to distinguish the difference between moving objects and irrelevant background motion which can be caused by the weather, illumination, camera movement etc. As foreground segmentation is often the first step of various highly important applications (video surveillance for security, patient/infant monitoring etc.), it is crucial to develop a model capable of producing excellent results in all kinds of conditions.

In order to tackle this problem, we follow the recent trend in other computer vision areas and harness the power of deep learning. We design architectures of convolutional neural networks specifically targeted to counter the aforementioned challenges. We first propose a 3D CNN that models the spatial and temporal information of the scene simultaneously. The network is deep enough to successfully cover more than 50 different scenes of various conditions with no need for any fine-tuning. These conditions include illumination (day or night), weather (sunny, rainy or snowing), background movements (trees moving from the wind, fountains etc) and others. Next, we propose a data augmentation method specifically targeted to illumination changes. We show that artificially augmenting the data set with this method significantly improves the segmentation results, even when tested under sudden illumination changes. We also present a post-processing method that exploits the temporal information of the input video. Finally, we propose a complex deep learning model which learns the illumination of the scene and performs foreground segmentation simultaneously.

Based on adversarial machine learning, the model comprises of six sub-networks, three generators and three discriminators. We show that jointly training the model to perform illumination change from bright to dark and vice versa, and foreground segmentation, yields substantial performance improvements over the state-of-the-art.

In conclusion, the proposed models address major limitations that currently exist in the area of background subtraction. As mentioned above, these models can be used in a plethora of closely related areas of computer vision and can have a practical impact as they can be embedded in real-world applications.

Table of contents

List of figures **xv**

List of tables **xxi**

Nomenclature **xxv**

1 Introduction **1**

1.1 Research Area 4

1.1.1 BGS using spatio-temporal (3D) convolutions 4

1.1.2 Synthetic data augmentation for robust BGS 5

1.1.3 BGS using triple-GAN to handle significant illumination changes . 5

1.2 Contributions 6

1.2.1 Publications 7

1.3 Thesis Structure 8

1.4 Definition of Terms 9

2 Background **11**

2.1 Gaussian Mixture Models 12

2.2 Kernel Density Estimates 14

2.3 Principal Component Analysis 17

2.4 Deep Learning 20

2.4.1 From classification to segmentation with fully convolutional networks 21
2.4.2 Dilated convolutions 25
2.4.3 Conditional Random Field for boundary refinement 26
2.4.4 Multi-task networks 28
2.5 Background Subtraction with Deep Learning 28
2.5.1 Single frame methods 28
2.5.2 Spatio-temporal models 30
2.5.3 Patch-based methods 34
2.5.4 GAN-based models 34
2.6 Conclusion 38

3 End-to-End Video Foreground Segmentation with 3D Convolutional Neural Networks 41

3.1 Introduction 42
3.2 Methodology 43
3.2.1 Network architecture 44
3.2.2 End-to-end training 46
3.2.3 Implementation 47
3.3 Evaluation 48
3.3.1 Datasets 48
3.3.2 Metrics 49
3.3.3 Results 50
3.4 Conclusion 59

4 Synthetic data augmentation for robust foreground segmentation 65

4.1 Introduction 66
4.2 Methodology 67

4.2.1 Local changes . . . 68
4.2.2 Global changes . . . 69
4.2.3 Combined changes . . . 71
4.2.4 Output refinement via temporal coherence . . . 73
4.2.5 Illumination-invariant Deep Networks . . . 74
4.2.6 Training settings . . . 75
4.2.7 Implementation details . . . 77
4.3 Evaluation . . . 77
4.3.1 Metrics . . . 77
4.3.2 Dataset . . . 78
4.4 Results . . . 78
4.4.1 Quantitative Evaluations . . . 79
4.4.2 Qualitative Evaluations . . . 82
4.4.3 Ablation studies . . . 82
4.4.4 Failure cases . . . 83
4.5 Conclusion and Future Work . . . 89

5 Illumination-aware Multi-task GANs for Foreground Segmentation 91
5.1 Introduction . . . 92
5.2 Methodology . . . 93
5.2.1 Pre-processing with gamma correction . . . 96
5.2.2 Feature extraction . . . 97
5.2.3 Foreground Fegmentation . . . 98
5.2.4 Training . . . 100
5.3 Results . . . 101
5.3.1 Evaluation Metrics . . . 101

5.3.2 Implementation details 102
5.3.3 SABS Dataset 102
5.3.4 ESI Dataset 106
5.3.5 Complexity comparison 115
5.3.6 Ablation studies 115
5.4 Conclusion 116

6 Conclusion and Future Work **119**
6.1 Summary 119
6.1.1 3D convolutions for spatio-temporal context 119
6.1.2 Semantic data augmentation for adapting to illumination changes . 120
6.1.3 Multi-task GANs for learning illumination changes 120
6.2 Review of Contributions 121
6.3 Directions for Future Work 121

References **123**

Appendix A Chapter 1 model **137**

Appendix B Chapter 2 model **155**

Appendix C Chapter 3 model **159**
C.1 Segmentation network 159
C.2 Discriminator 161
C.3 VGG and Generator 161

List of figures

2.1 Gaussian (parametric) distribution, Gaussian Mixture Model, and Gaussian (kernel) density estimator based on 20 samples generated from the mixture of uniform distributions: $p_X(x) = \frac{1}{2} \times U(x;-8,-1) + \frac{1}{2} \times U(x;1,8)$, where $U(x;a,b) = \frac{1}{(b-a)}$ denotes the continuous uniform probability density function for random variable X. Figure taken from [119]. 14

2.2 The framework of a PCA-based background subtraction method. The background image is reconstructed using the eigenbackground which correspond to the most significant eigenvectors. Then, the foreground can be recovered by subtracting the current frame to the reconstructed background image. Image were taken from [44]. 18

2.3 The effect of using a max/average pooling and unpooling layers. The different colours represent the movement of a 2x2 kernel sliding across the 4x4 input with a stride of 2. During pooling, the position of the maximum activation is saved using an index mask. In unpooling, this mask is used to place the activations of the deep feature map to their original position. . . . 22

2.5 A typical Convolutional Neural Network with an encoder-decoder architecture for background subtraction. The encoder is consisted of multiple Convolution layers which extract semantic features, followed by Pooling layers for dimensionality reduction. The decoder uses Upsampling layers to restore the resolution of the feature maps to its original size, while Convolution layers are used to recover the information lost due to downsampling. The ReLu and Batch Normalisation layers ensure the stability of the training process, while the Softmax layer converts the features of the last layer to pixel-wise probabilities. Parts of this image were taken from [6]. 24

2.6 An overview of PSPNet with the groundbreaking pyramid pooling module. The output of the last convolutional layer is fed to the module, which uses pooling layers with kernels of various sizes to extract features from different scales. The output is then upsampled and concatenated. [161] 26

2.7 Applying a Conditional Random Field (CRF) to the Deep Convolutional Neural Network (DCNN) output vastly improves the boundaries of the segmentation mask. Image taken from [20]. 27

2.8 Visualisation of a Long Short-Term Memory unit, a type of recurrent neural network for modelling long term patterns. The symbols $\otimes$ and $\oplus$ denote point-wise multiplication and addition respectively, whereas σ is a sigmoid function. Finally, x_i and h_i are the unit's input/output. Image taken from [90]. 32

3.1 The network architecture. The input comprises a video of 10 frames which is connected to the first group of layers, CRP1, in groups of 4 frames with stride 2. CRP1 is then connected to CRP2 in the same manner and CRP3 has access to the features of all frames. CRP4 is performing 2D operations only, while CR has no pooling layer. The upsampling layers US1, US2, US3 and US4 are connected to CRP2, CRP3, CRP4 and CR respectively and are concatenated before applying the final convolution. The full layer specifications are presented in table 3.2. It should be noted that cubes indicate 3D operations across the temporal dimension, while rectangles indicate 2D (spatial only) operations. The plus sign indicates concatenation. 45
3.2 ROC curve on CDnet dataset 55
3.3 ROC curve on ESI dataset 58
3.4 F-Measure values comparison on the ESI dataset [1] 59
3.5 Segmentation results of ESI [135], MBS [106] and ours on ESI dataset. [2] . 60
3.6 Segmentation results of our model in all categories of the CDnet dataset. . 61
3.7 Failure cases. Pixels in grey colour are outside of the region of interest. . . 62
3.8 Failure cases. Pixels in grey colour are outside of the region of interest. . . 63

4.1 The application of the mask for local changes. Subfigure (a): the initial binary mask M_1 is created by a circle of diameter $d = 179$ and centre coordinates $(322, 265)$. Subfigure (b): The mask M_2 after the application of the Euclidean distance transform on M_1. Subfigures (c) and (d) depict the original image and the lamp-post effect after the application of the mask M_2 on the input image respectively. 70
4.2 Methodology 71

4.3 Combination of global and local illumination changes. The subfigures (a) and (b) depict a combination of a brightening global filter with a bright and dark local filter respectively. On the other hand, subfigures (c) and (d) implement the darkening global filter. 72

4.4 The probability mask that is used for refining the output, created from the segmentation mask of the previous frame. Bright colours indicate high probability, whereas dark colours show low probability values. 74

4.5 The CNN that was used for the experiments. The encoder is initialised from VGG16 [115] and is keep fixed during training. ReLu layers are used after every convolution and are omitted from Figure (a) for clarity. Features of the encoder are concatenated with those of the decoder which are of the same size, to enable information flow. 76

4.6 Regular augmentation techniques (from left to right): image mirroring, center cropping and adding noise. 79

4.7 F-Measure values on different thresholds for each model. 84

4.8 The SABS dataset that was used for evaluating the models. The first row depicts the training sequence *Darkening*, while the second row shows the testing video *LightSwitch*. The columns show frames from the start, middle and ending parts of the video. Note that in the middle of the *LightSwitch* sequence the store light switches off, causing major changes to the background. 85

4.9 Failure cases 86

4.10 Results with various kernels and intensities for the application of local changes. 87

4.11 Comparison between different augmentation techniques. 88

5.1 The architecture of TMT-GAN. Given an image, a low/high brightness image pair is generated with the gamma function. Both images individually undergo feature extraction by $VGG16$, followed by image generation by G_b and G_d. Red arrows show the path for bright images and blue arrows for dark images. Multi-scale features are extracted from three inner layers of G_b and G_d and incorporated into the foreground segmenter G_s with pixel-wise subtraction. D_b, D_d and D_s are the discriminators of the respective generators and ensure the distribution matching between the real and generated images. The plus and minus signs indicate feature concatenation and element-wise subtraction respectively. The ground truth construction is explained in section 5.2.2. . . 95

5.2 The attention module 99

5.3 Consecutive frames in the SABS and ESI datasets featuring sudden illumination changes 103

5.4 Comparison of F-Measure values with state-of-the-art models on the *Light Switch* sequence of SABS. Statistics are taken from Shimada and Taniguchi [114]. 106

5.5 Comparison of F-Measure values with state-of-the-art models on the *Light Switch* sequence of SABS. Statistics are taken from Javed et al. [55]. 107

5.6 Qualitative results on the SABS dataset 107

5.7 Failure cases 108

5.8 Qualitative results on the ESI dataset 114

List of tables

2.1 Summary of background subtraction/foreground segmentation methods . . 39

2.2 Summary of Deep Learning - based background subtraction/foreground segmentation methods 40

3.1 Overall statistics across all categories on CDnet2014 for the top 10 models [3] 51

3.2 In-depth look of our network architecture. Each row represents the full specifications of each group of layers, as depicted vertically from left to right in figure 3.1. CRP: Convolutional-ReLu-Pool, CR: Convolutional-Pool, US: Up-Sampling, FC: Final-Convolution 52

3.3 Results on unseen videos 53

3.4 Category-wise results on CDnet dataset 54

3.5 F-Measure results on the ESI dataset 58

4.1 The different augmentation settings that were tested in our experiments. Parameters k, z and X denote the kernel size of the mask M_1, the illumination intensity in terms of pixel values and the resolution of the smallest dimension of the input image respectively. The last column shows the threshold that binarises the model output and which maximised the F-Measure of the segmentation mask. The GL_{refine} model employs the post-processing method described in Section 4.2.4. 80

4.2 Comparison between no augmentation, common augmentation and the proposed method which covers global and local illumination changes. Best scores are highlighted in bold. 81
4.3 Ablation studies for local and global changes 84

5.1 Scenes and frame indices used in training and testing on the SABS dataset . 103
5.2 Scenes and frame indices used in training and testing on the ESI dataset . . 109
5.3 Results on the SABS (Light Switch) dataset. Best scores are highlighted in green colour, while second best are depicted in red font. 110
5.4 Results on the SABS (Light Switch) dataset. Best scores are highlighted in green colour, while second best are depicted in red font. 110
5.5 Results on the ESI dataset by category 111
5.6 Results on the ESI dataset by category 111
5.7 Results on the ESI dataset 112
5.8 Results on the ESI dataset 112
5.9 F-Measure values of the model trained on SABS if a component is removed 113
5.10 F-Measure values of the model trained on ESI if a component is removed . 113
5.11 Analysis on the parameter count of the evaluated models 115

Nomenclature

Roman Symbols

BGS Background Subtraction

CNN Convolutional Neural Network

CRF Conditional Random Field

CRP Convolution - ReLu - Pooling

EDT Euclidean Distance Transform

ESI Eigenbackground Statistical Illumination

FCN Fully Convolutional Network

FgSegNet Foreground Segmentation Network

FM F-Measure

GAN Generative Adversarial Network

GMM Gaussian Mixture Model

KDE Kernel Density Estimate

LBP Local Binary Pattern

LSTM Long Short-Term Memory

OSVOS One-Shot Video Object Segmentation

PCA Principal Component Analysis

RPCA Robust Principal Component Analysis

SABS Stuttgart Artificial Background Subtraction

STN Spatial Transformer Network

$TMT-GAN$ Triple Multi-Task Generative Adversarial Network

Chapter 1

Introduction

In the domain of video processing, foreground segmentation (alternatively background subtraction) is the process of discriminating moving objects, defined as foreground, from static parts of a given scene, or background [8]. This task can be regarded as a binary pixel-wise classification with the label taking the values of either background or foreground.

This research area has sparked a growing interest among scientists for more than 30 years. However, as the camera technology advances year by year, the videos increase in complexity and size, but also in numbers. Thus, researching new background subtraction algorithms, able to handle big data efficiently and effectively, is more crucial than ever. Recently, after the flagrant successes of deep learning in big data processing in images [115, 67, 48, 150], neural network models emerge in the spotlight across the research community. Thus, it is time the angle in which foreground segmentation is viewed to change and the algorithms structure to be reconsidered.

The significance of innovative background subtraction can be highlighted by a plethora of real-world applications. Specifically, it has been used for video surveillance related tasks like crowd analysis, abnormal event monitoring, intelligent traffic control systems, people safety and animal surveillance in the wild [64]. Other areas of application include person

re-identification [9], object tracking [113], gesture recognition [151], vehicle tracking [116], video recognition [68], crowd analysis [139] and even use cases of the medical domain [129]. It is evident that many of those are critically important, and even the slightest increase in detection accuracy is crucial. A major hurdle is to design algorithms able to capture the significant changes in the video frames, while ignoring noise-produced changes. Such changes could be produced by a variety of different factors [64, 5, 106, 57, 13]:

- Illumination: Depending on the time the video was captured, it is possible that a significant amount of pixels have changed due to lighting conditions, caused either by the sun or artificial light. This is especially evident in sudden illumination changes that might occur if a light switch is turned on/off, or if a large cloud blocks the sun. In addition, a scene and the objects that appear in it will drastically transform during the night. It is necessary for an algorithm to be able to adjust in this kind of conditions.

- Weather: A scene will appear different under rain or snow. While the sun causes objects to appear brighter, rain grants them a gray shade and snow paints them white. As a result, special attention is needed to avoid false classifications.

- Background motion: Most scenes include moving objects which still need to be classified as background. Such objects are trees, fountains and bodies of water in general.

- Shadows: Some background objects might cast shadows in scenes taken under intense sunlight. Those pixels affected will be darker than usual and as a result are prone to misclassification.

- Intermittent object motion: There is a special case of videos where various objects will move into the scene and proceed to stay still for an indefinite amount of time. Examples include parked cars, people walking and sitting, various objects being thrown etc. It

is obvious that the correct classification of these objects is entirely case-dependent. However, in any case, the model needs to adjust accordingly.

- Camera angle: It is not unusual for several cameras to change angle while filming, causing the background to change. This scenario is especially prevalent in video surveillance, where the cameras are used to rotate in order to capture a larger background area, and sometimes zoom in and out. These videos are exceptionally troublesome to analyze and novel ideas are needed to achieve high accuracy.

Of course in many cases, changes occur by a combination of the elements listed above. Naturally, the more factors are involved, the more difficult it becomes to extract the foreground accurately.

Most traditional approaches employ a 3-step approach: Background initialization, foreground detection and background maintenance [118]. First, the background is composed using a number of frames and then the segmented image can be obtained by comparing the current frame to the background. The third step is devoted to updating the background model with the most recent changes of the background. The main problem of these methods lies in their inability to obtain good representations of the input data. As a result, when the background model is challenged with significant changes, the foreground detection module produces many misclassifications. On the other hand, deep learning models are able to approximate extremely abstract, non-linear functions by stacking a large number of layers on top of each other. Such models can extract high level features of a scene and therefore produce better segmentation results. For example, given enough input frames with trees in the background, a deep learning model will uncover the patterns between the pixels corresponding to the tree's leaves, branches and trunk and develop an abstract representation of the tree. Therefore, even if the tree is moving because of a strong wind, the model will still classify those pixels as background.

1.1 Research Area

As mentioned above, there are several challenges remaining to be addressed in the research area of foreground segmentation. To achieve robust background subtraction which addresses all aforementioned limitations, we propose the following deep learning models:

1.1.1 BGS using spatio-temporal (3D) convolutions

Most conventional approaches -deep learning or not- operate in a frame-by-frame basis. However, a video is not a mere collection of individual frames, in the sense that each frame is heavily linked to the previous and next ones. Therefore, predicting a segmentation mask based only on the current frame means missing valuable information. Although some models retain a history of past frames, either as a mean frame or as a background model, they are still unable to model the continuity of the pixels between adjacent frames. This is especially important in the case of sudden background changes, where the current frame is considerably different compared to any previous frames. Rather than treating a single frame as a separate entity, we propose a system that considers a frame as a part of a whole, and instead operates in a collection of contiguous frames. By performing convolutions with 3D kernels, we can effectively model the frame continuity of a video. Such an approach not only captures the relationships between spatially and temporally adjacent pixels, but it also renders a background model obsolete. A system of this nature is inherently more accurate for two main reasons:

1. It no longer depends on the soundness of the background model and its update process

2. The spatial and temporal changes between frames are captured in the higher, non-linear dimensional space which contains rich semantic information.

Furthermore, we show that it is possible to model multiple (over 50) videos with a single model, whereas traditional models can only monitor a single scene.

1.1.2 Synthetic data augmentation for robust BGS

Having covered most cases with our previous model, we proceed to address the most challenging issue in foreground segmentation: intense and sudden illumination changes. As the appearance of all objects changes drastically under different illumination conditions, even deep learning models struggle to cope in such cases. This is especially apparent in the "light switch" scenario, where the source of light is turned off immediately. If the lighting is very limited, the model will be unable to extract semantic information from the scene.

To overcome this hurdle, we propose a novel image augmentation method that generates synthetic images by altering the illumination of the input not only in local areas but also globally. Essentially, we provide extra semantic information to the model in terms of illumination, which leads to better generalisation in scenes depicting light-based effects such as halos and shadows. We show that this approach yields significantly better results compared to traditional augmentation techniques, when tested in data sets that feature changes of brightness.

Further improvements can be obtained by applying a post-processing method that takes advantage of the temporal information of the input video. In particular, we exploit the limited amount of movement of the foreground objects between adjacent frames, in order to filter noise caused by false positives in some parts of the image.

1.1.3 BGS using triple-GAN to handle significant illumination changes

Next, we attempt to learn the illumination of the scene, rather than artificially changing the brightness of the input image. In particular, we propose a triple *Generative Adversarial*

Network to model the scene at different illumination scales. The model is trained to perform BGS when given extremely bright and dark images, and also to learn the illumination of the scene by converting bright images to dark and vice versa, while keeping a focus on the foreground. In order for the generated images to look realistic, the model is forced to not only learn the difference between the foreground and background classes, but also to learn to change the illumination of the scene. Essentially, the objective is three-fold:

1. Obtain rich semantic features of the objects appearing in the scene regardless of the lighting

2. Model the illumination of the scene by using GANs to change the lighting of both foreground and background objects

3. Perform robust BGS using the features which are extracted from the GANs.

We show that this approach is superior even to state-of-the-art deep learning models.

1.2 Contributions

The contributions of this thesis can be summarized as follows:

- We propose a 3D Convolutional Neural Network for foreground segmentation to effectively model spatial and temporal information. The model is an end-to-end, standalone system and is can handle multiple scenes.

- We propose a novel image augmentation method which alters the illumination of the input in local areas but also globally. We also propose a post-processing technique for noise removal. The effectiveness of both methods is demonstrated by training a deep neural network with the augmented dataset.

- We propose a triple Generative Adversarial Network for foreground segmentation, which is able to analyse the illumination conditions of the scene. The model produces accurate segmentation results even in a "light switch" scenario, which features insufficient lighting and rapid changes of the background.

1.2.1 Publications

Portions of the work presented in this thesis have previously been published in the following academic papers:

- Dimitrios Sakkos, Edmond S. L. Ho and Hubert P. H. Shum, "Illumination-aware Multi-task GANs for Foreground Segmentation", *IEEE Access*, Jan 2019.
  Bibliography reference [107]

- Dimitrios Sakkos, Heng Liu, Jungong Han and Ling Shao, "End-to-end video background subtraction with 3D convolutional neural networks", *Multimedia Tools and Applications*, Sep 2018.
  Bibliography reference [109]

- Dimitrios Sakkos, Edmond S. L. Ho and Hubert P. H. Shum, "Synthetic data augmentation for robust background subtraction", *SKIMA 2019*, Under review.
  Bibliography reference [108]

- Daniel Organisciak, Dimitrios Sakkos, Kateřina Jandová, Edmond S. L. Ho, Nauman Aslam and Hubert P. H. Shum, "Unifying Person and Vehicle Re-identification", *IEEE Transactions on Circuits and Systems for Video Technology*, Under review.
  Bibliography reference [93]

1.3 Thesis Structure

In Chapter 2, we provide an extensive literature review for the task of background subtraction. Both traditional and state-of-the-art approaches are discussed and analysed. We also review some techniques on general image segmentation.

In Chapter 3, we propose a 3D convolutional neural network for video foreground segmentation. We analyse the network architecture and explain the function and contribution of each module. The experimental results show that the model learns to extract the foreground objects in great accuracy, with very sharp object boundaries. In addition, the results show that it is possible to use a single model for over 50 different scenes, each featuring different challenges.

In Chapter 4, we propose an augmentation method that can generate an infinite amount of unique synthetic images by performing local and global illumination changes. We show that these semantic changes are superior to common augmentation techniques. In addition, a temporally-based post-processing method is proposed for noise removal. The experiments indicate that this method can further improve segmentation results.

In Chapter 5, we propose an illumination-aware convolutional neural network which is comprised of three GANs. The design choices of the network are discussed and explained. The model is tested on the most challenging public datasets which feature rapid and intense illumination changes. The experimental results show that the network can handle even a "light switch" scenario and produce robust segmentation masks.

In Chapter 6, we provide a conclusion of this thesis and discuss the potential future work on this research area.

1.4 Definition of Terms

Deep learning is a sub-category of machine learning which is concerned with the development of neural networks with a large number of layers; otherwise known as deep neural networks. The deeper a network is, the more complex functions it can learn.

Convolutional neural network is a special kind of neural network that is inherently designed to handle images. Unlike fully connected layers, convolutional layers do not have connections between all neurons. Instead, they connect each neuron only to a small part of the input, based on the intuition that neighboring pixels share common attributes.

Generative adversarial network is a special king of convolutional neural network that deals with the generation of fake images. They are consisted of two neural networks: the *generator*, which is responsible of generating images, and the *discriminator*, which is trained to distinguish real from fake images. By jointly training the two networks, it is possible to generate very real-like images.

Feature map is the output of a convolutional layer and it embodies an abstract representation of the input data.

Chapter 2

Background

Over the years, a wide variety of approaches have been employed to automatically extract the foreground from a given video. Early techniques used simple methodologies like frame differencing or subtracting the current frame from a background model that was computed by averaging a set of past frames. However, such simplistic approaches were successful only when certain criteria were met. For example, they were unable to capture slow-moving foreground and would fail in videos with intermittent object motion [38]. More advanced approaches learn meaningful spatio-temporal features from videos [77, 149, 163]. Some methods tackle challenges mentioned in Chapter 3.1 by employing preprocessing techniques in the input data, before feeding it to the model [26]. Furthermore, other approaches include post-processing steps [15] in an attempt to refine the model output and improve the segmentation accuracy. In this Chapter we categorize the most significant existing methods based on their approach and analyse their advantages and limitations.

[1] Average taken for indoor and outdoor video results

2.1 Gaussian Mixture Models

One of the most widely researched algorithm types for background subtraction is based on a mixture of Gaussian distributions. A major advantage of this model family is that it can be performed in an unsupervised manner, therefore there is no need for labelled data. First proposed by Friedman and Russel [38], the scene of a video sequence was modeled by learning a mixture of a Gaussians with a fixed number of components for each pixel of the input data. Since the model parameters are updated according to the probability of each pixel belonging to a specific class, slow-moving objects are segmented accurately. There is a necessity to have multiple gaussians, so that different classes can be effectively monitored. This approach used three gaussians, for the following classes: foreground, shadows and background. Zivkovic [170] extended this approach by adding more gaussians to the mixture model, the specific number of which was automatically estimated and updated for each frame of the video. Therefore, the model was able to monitor the scene more effectively. Siva et al. [117] extend this method by combining GMM with a conditional probabilistic function, in an attempt to tackle illumination changes more effectively. More specifically, they use the GMM to obtain a good background image, which is then used by the probabilistic model that classifies the foreground pixels. In contrast to previously developed models which map the current frame pixels to background image pixels in a one-to-one mapping, the probabilistic model is able to map different pixel intensities to the same background intensity, therefore catching illumination changes more accurately. Recently, Boulmerka and Allili [12] combined a GMM with statistical models. The GMM was used to model the temporal information, whereas inter-frame correlation analysis and histogram matching were employed to capture the spatial relationship of adjacent pixels. A median filter was applied for noise removal as a post-processing step, after combining the masks of the two models. While this approach works well in dynamic environments and is able to remove shadows, it

depends on many different methods. Thus, the parameter fine-tuning process but also the deployment of the model can be cumbersome.

Akilan et al. [2] also combine a GMM with several other methods to obtain better results. The GMM is again used for the estimation of the background image, where a varied version of the Bhattacharyya distance is used to determine if a pixel matches the GMM. The foreground is then obtained by subtracting the current frame to the background image. Consequently, the result is enhanced by fusing features of color similarity, color distortion, and illumination measures which are obtained from the current frame. Chen et al. [23] use a number of GMMs to construct spanning trees for hierarchical superpixel segmentation. They report that extending their model with optical flow for modeling temporal information increases the segmentation accuracy. While the superpixel hierarchy method removes false positives effectively in dynamic backgrounds, it is unsuccessful at removing shadows. Shen et al. [113] propose an efficient approach to BGS by reducing the dimensionality of the input data with a random projection matrix. Finally, they apply a GMM on the projected data. Although GMM-based methods perform well on videos with minimal or gradual illumination changes, they fail when challenged with rapid variations of illumination [118]. Pilet et al. [98] made an attempt to rectify this issue by modelling the illumination ratio of each pixel, rather than its intesity. In more detail, they assume each pixel p_i value can be decomposed to $p_i = e_i a_i$, where e_i the irradiance and a_i the albedo of the object depicted by the pixel. Therefore, if the same property is true for the model image m, we have $m_i = e_m a_i$, with the irradiance e_m now being constant. As a result, the illumination ratio is $l_r = \frac{u_i}{m_i} = \frac{e_i}{e_m}$. Having created two GMMs for modelling the ratios for the background and foreground, they use a spatial likelihood model to learn the relationship between the model output and the target segmentation. Finally, Tuzel et al. [132] modeled different representations of each pixel with a respective layer of 3D multivariate Gaussians, resulting in a multilayer Gaussian mixture model. Their background model was updated gradually, by changing at most one layer

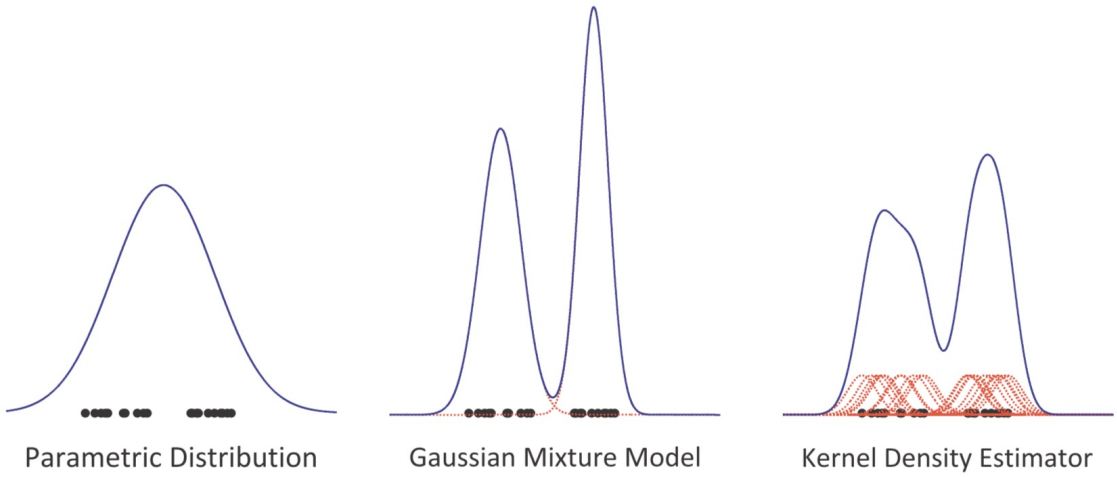


Fig. 2.1 Gaussian (parametric) distribution, Gaussian Mixture Model, and Gaussian (kernel) density estimator based on 20 samples generated from the mixture of uniform distributions: $p_X(x) = \frac{1}{2} \times U(x;-8,-1) + \frac{1}{2} \times U(x;1,8)$, where $U(x;a,b) = \frac{1}{(b-a)}$ denotes the continuous uniform probability density function for random variable X. Figure taken from [119].

on every update. With such a multimodal approach the model performed well in dynamic backgrounds.

While a lot of research has been done on improving Gaussian Mixture Models (GMMs) in dynamic environments, their disadvantage on adapting in sudden lighting changes and heavy shadows remain [64]. Another limitation is that they operate in a pixel level and do not model the correlation between adjacent pixels, therefore missing important information.

2.2 Kernel Density Estimates

The Kernel Density Estimates (KDE) models follow a different probabilistic, non-parametric approach for scene modeling which addresses the issue of parameter estimation and update in GMMs. The probability density function of a KDE is given as below:

$$\hat{f}(y) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{x_i - y}{h}\right),$$

where d is the dimensionality of the higher dimensional space,

x_i is an independent and identically distributed random variable of size n, $x_i \in \mathbb{R}^d$,

$K : \mathbb{R}^d \to \mathbb{R}^1$ is a function centered at 0 that integrates to 1,

h is a smoothing parameter which tends to 0 as n tends to inf,

n denotes the sample size.

An illustration of the construction of a KDE model versus a GMM is given in Figure 2.1.

Elgammal et al. [36] implemented a real-time model which estimates the probability density function of a pixel's intensity values by observing a fixed size window of the most recent frames. Their model used the Normal function $N(0,\Sigma)$ as the kernel estimator function. Because this method is very sensitive to false positives in dynamic backgrounds, the authors impose spatio-temporal constraints to the model in an attempt to suppress false detection: First, they estimate the pixel displacement probability, based on the premise that only minor changes will occur in a small number of frames and secondly, they calculate the component displacement probability, which hypothesises that if an object belongs to the foreground, it will have moved from a nearby location. Thus, the output is refined.

There are also methods that are using kernels of variable size. In general, there are two ways of varying the kernel width [127]:

1. *The balloon estimator*, where the kernel size is a function of the point y at which the estimate is taken:

$$\hat{f}_1(y) = \frac{1}{nh(y)^d} \sum_{i=1}^{n} K\Big(\frac{x_i - y}{h(y)}\Big) \tag{2.1}$$

2. *The sample smoothing estimator*, where the kernel size depends on the sample point x_i:

$$\hat{f}_2(y) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h(x_i)^d} K\Big(\frac{x_i - y}{h(x_i)}\Big) \tag{2.2}$$

Zivkovic and Van Der Heijden [171] modify the traditional approach by employing a kernel of varying size at each estimation point, therefore using the balloon estimator. Additionally, a decaying factor was added in the update mechanism of the model such that older samples have lower influence. It was found that the model is superior to traditional GMMs in scenes with dynamic backgrounds. Mittal and Paragios [89] implement a hybrid density estimator which is combination of the two aforementioned estimators. Specifically, the hybrid density estimator is defined as follows:

$$f_H(x) = \frac{1}{n}\sum_{n=1}^{n} \frac{1}{||H(x,x_i)||^{\frac{1}{2}}} K(H(x,x_i)^{-\frac{1}{2}}(\mu - \mu_i)) \tag{2.3}$$

where $H(x,x_i) = \sum_{x_i} + \sum_x$ is the bandwidth matrix for the *Normal* kernel.

The authors also employed a dynamically adjusted threshold for the pixel classification, which helps remove false positives in dynamic backgrounds. Finally, they incorporated optical flow information and colour normalisation in an attempt to capture motion information and make the model invariant to illumination changes respectively. Attempts have also been made for improving the model accuracy with post-processing operations. For example, Zhu et al. [169] implemented a method that addresses the inability of pixel-level techniques to model the spatial relationship of neighbouring pixels. Specifically, the authors refined the segmentation mask with a local texture correlation operation that fuses the missing information from the input image. Finally, Berjón et al. [10] utilise temporal information, as previous frames are selected to guide the update of the background model. Then, the outputs of the foreground and background model are fed to a bayesian classifier which performs the background subtraction.

In conclusion, the KDE models can address some of the limitations of GMMs and the literature suggests they are more effective in videos with dynamic backgrounds. However,

these models still cannot inherently capture the relationship of adjacent pixels and need to rely on post-processing techniques.

2.3 Principal Component Analysis

The methods of this category aim to construct a background model by decomposing the input frames into a low-rank subspace, which is constructed using eigenvectors. Since PCA retains the most significant eigenvectors, the foreground of the input image cannot be represented by the background model, as long as it is not static. The foreground can then be segmented with a difference image between the output of the model and the input frame [91]. Essentially, the foreground pixels are being detected as outliers using a threshold [13]. A typical pipeline of such a BGS model is given in Figure 2.2.

One of the first methods was implemented by Oliver et al. [91] who constructed the mean background image and its covariance matrix using N frames. Principal Component Analysis (PCA) was performed to reduce the dimensionality of the space. However, this approach does not perform well in cases where background pixels constantly change (due to weather / light conditions or change of camera angle) or foreground pixels remain unchanged (intermittent object motion), since it is based on the idea that the moving objects will appear in different parts of every frame in the sample.

These limitations where addressed with Robust PCA (RPCA) [130, 18], which revolutionized PCA-based methods. By decomposing the input data matrix A into a low rank matrix L and a sparse matrix S such that $A = L + S$, the background and foreground are successfully modeled by L and S respectively. This process is called RPCA via Principal Component Pursuit (PCP). Because L is updated over time, the model can adapt to challenging environmental conditions more effectively. However, RPCA-PCP is computationally expensive, can only be batch updated rather than frame by frame and is also sensitive to noise

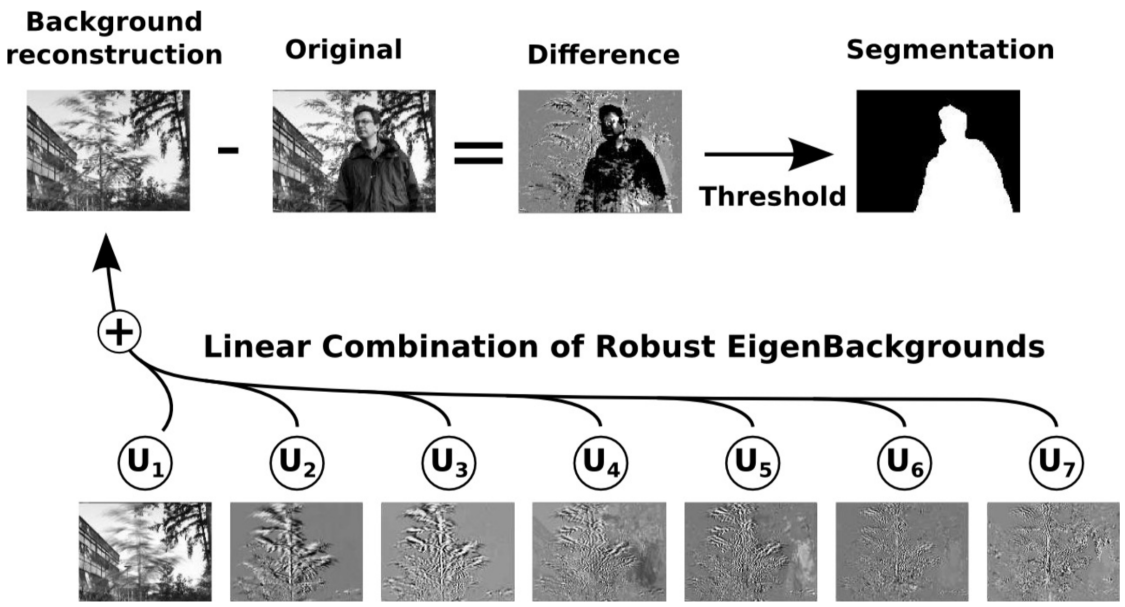


Fig. 2.2 The framework of a PCA-based background subtraction method. The background image is reconstructed using the eigenbackground which correspond to the most significant eigenvectors. Then, the foreground can be recovered by subtracting the current frame to the reconstructed background image. Image were taken from [44].

[13]. Liu et al. [78] addressed the first limitation and proposed an improvement of the PCP algorithm in terms of efficiency. Specifically, they solve PCP in linear time by reformulating the decomposition of the input matrix in a manner than can be run in parallel. Zhou and Tao [166] implemented an approximate solution for RPCA that can handle noisy cases. Instead of using Singular Value Decomposition for decomposing the input matrix A into L and S, they speed up the calculations by using Bilateral Random Projections. The input matrix is then decomposed into three matrices L, S and G such that $A = L + S + G$, where G is used to represent the noise.

Zhang et al. [159] developed a background subtraction method which is highly resilient to dynamic scenes, since the threshold that is used for the foreground segmentation is calculated dynamically. In addition, the correlation between neighboring pixels is successfully modeled using local dependency histograms. A different approach was implemented by Zhang et

al. [158], who extended the classic Local Binary Patterns (LBPs) to the spatio-temporal domain. Specifically, they compute an LBP over the two most recent frames which effectively tracks both spatial and temporal information. Zhang et al. [160] devised a Covariance-based technique for modeling a scene. Specifically, the background of a region around each pixel is represented by a covariance matrix constructed by features based on pixel coordinate and intensity values as well as LBPs. The foreground is then detected by comparing the new covariance matrix versus the current background matrix using a threshold.

Candès et al. [18] developed an efficient algorithm (RPCA) for decomposing the data into a low-rank matrix and a sparse matrix, which are representing the background and foreground in the BGS scenario, respectively. Recently, Ibadi and Isquierdo [34] extended RPCA by using a tree-structured sparse matrix to represent the input images. Although their method performs well on standard datasets, it fails in videos with sudden illumination changes like the *Light Switch* sequence of the *SABS* dataset. Xin et al. [145] also extended RPCA by utilising contextual information of the foreground pixels with the generalized fused lasso regularization which was originally proposed in [144]. Although they report the SABS dataset, only the *basic* video sequence was used, which has negligible changes in illumination. The pixels of the shadow were incorrectly classified as foreground, as the difference in the illumination makes them darker. Similar to Pillet et al. [98], Vosters et al. [134] extend traditional PCA-based models by introducing a statistical illumination model. Basically, they create an eigen-space background model which is used to reconstruct the background of the current frame by projecting it to the space of the learned model. However, instead of using a simple thresholding technique for the final segmentation, the authors develop a spatial likelihood model that improves the segmentation results by modelling the relationship of neighboring pixels. In contrast with [98], the algorithm is updated online for each frame in order to adapt to rapid illumination changes. Therefore, the eigen-space model captures global illumination changes, while the spatial likelihood model models the

remaining background variations. In general, PCA-based methods are highly robust [64] since they analyze pixel changes in the lower dimensional space, however they fail to analyze multiple dissimilar scenes due to the data representation not being multi-modal as well as the high computation costs[13]. Additionally, although they are more robust to illumination changes than GMMs, they are limited by the lack of semantic knowledge in the scene. Finally, they cannot distinguish shadows, since they are considered a part of the foreground object and as such, they are not permitted in the training set for the development of the background model [135].

2.4 Deep Learning

With Deep Learning dominating the research area of Artificial Intelligence, it is now possible to develop an end-to-end system for image segmentation/background subtraction that addresses the vast majority of the challenges to a large extent. The most prominent type of neural networks in Computer Vision, Convolutional Neural Networks (CNNs), are achieving state-of-the-art results in numerous tasks, like object detection [47], person re-identification [70], face recognition [111], speech recognition [101], natural language processing [29], video classification [63] and others. Therefore, there is strong motivation to employ CNNs for background subtraction.

In this Section we will describe the building blocks of CNNs and analyse their inner workings. We will also discuss the evolution of CNNs through landmark papers and illustrate successful architectures. Finally, we will outline prominent post-processing methods.

2.4.1 From classification to segmentation with fully convolutional networks

Deep learning approaches for background subtraction use variants of the fully convolutional network (FCN) proposed by Long et al. [82], which was based upon the works of Sermanet et al. [112], Pinheiro and Collobert [99] and Eigen et al. [35]. This is a special kind of convolutional neural networks with no fully connected layers, specifically designed for dense prediction tasks like image segmentation. Using the very successfull network VGG [115] as a backbone, the authors transformed all fully connected layers into convolutional layers. As a result, FCN can be trained in an end-to-end manner and is much more efficient than patch based models. The network is extracting features from the input image through a succession of convolutions with different (trainable) kernels. However, since these operations can be computationally expensive, the dimensionality of the feature maps is gradually downsampled with a series of pooling layers, the function of which is depicted in figure 2.3. These layers employ either the max or the average operation using kernels, although the most commonly used pooling layer is the max pooling. Instead of max-pooling, learnable subsampling could be used instead, however it is shown to provide worse results [110]. In addition to computational efficiency, pooling layers also expand the receptive field of the output layer. Since each neuron of a CNN's layer is only connected to a small area of the previous layer, it has a limited receptive field. However, after a pooling layer, each pixel of the new feature map is a summary of a small neighbourhood of the previous. This means that the neurons of the deeper convolutional layer has access to information of a larger neighbourhood of the input. This is essential for all computer vision tasks, since it allows deeper layers to assimilate contextual information about the objects appearing in the input image.

The FCN has 5 max-pooling operations, which means that the output of the decoder is 32 times smaller than the input image. The full-sized output is then obtained by feeding

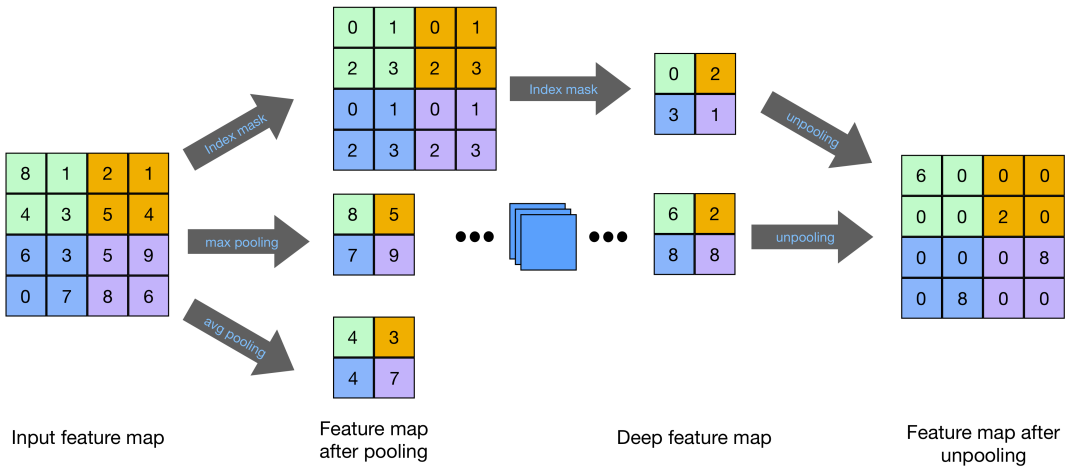
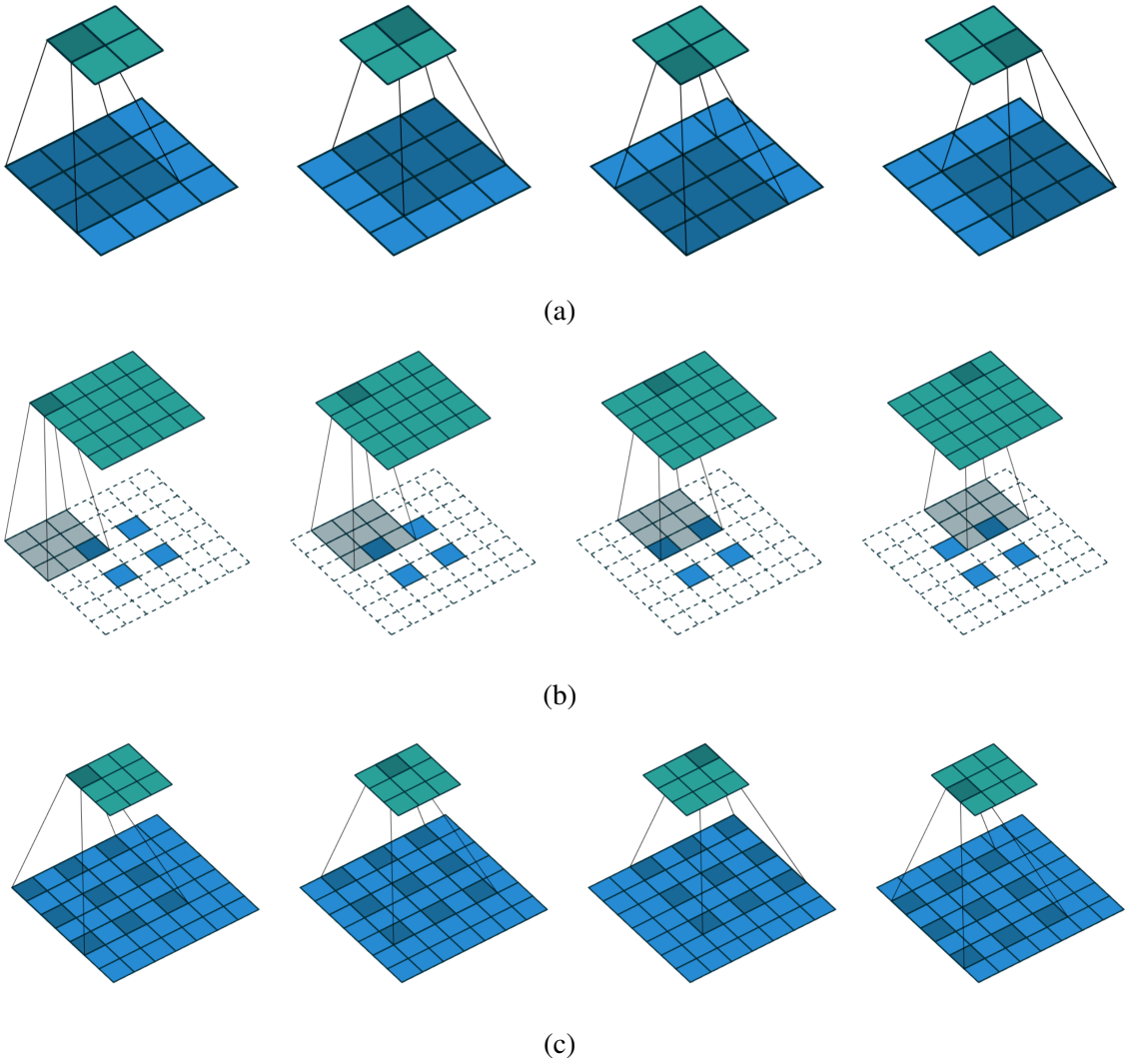


Fig. 2.3 The effect of using a max/average pooling and unpooling layers. The different colours represent the movement of a 2x2 kernel sliding across the 4x4 input with a stride of 2. During pooling, the position of the maximum activation is saved using an index mask. In unpooling, this mask is used to place the activations of the deep feature map to their original position.

the downsampled feature maps to the decoder, which reinstates the original size through a series of upsampling layers. These layers can be either unpooling or transposed convolution (deconvolution) layers; in the case of FCN deconvolution layers are used. Although the effect is similar, those layers are very different in the way they function. Transpose convolution works by inserting zeros between the pixels of the input feature map and then performing regular convolution (figure 2.4b). Since the weights of the kernel are learned, this method has the advantage of learning class-specific features during upsampling. Unpooling layers are, as the name suggests, reverse pooling. Essentially, the unpooling layer retains the spatial location of the maximum activation during pooling and places the same value at the corresponding index of the output feature map, filling everything else with zeros (figure 2.3. In contrast to transpose convolution, the position of the retained activations are the same as before downsampling, making this method better at preserving object boundaries.

Although the use of pooling layers has many advantages, the accumulated loss of information ends up being excessive and unrecoverable by the decoder. To alleviate this problem,



(a)

(b)

(c)

Fig. 2.4 Demonstration of (a) the convolution, (b) transpose convolution and (c) dilated convolution operations. Convolving a 4x4 input with a 3x3 kernel leads to a 2x2 output feature map. At the transpose convolution example, a 2x2 input feature map is padded with zeros in between the pixels, as well as the border. The number of padding depends on the desired output size. In this case, a 3x3 kernel leads to a 5x5 output feature map. Finally, in a dilated convolution the kernel is padded instead of the input, leading to an increased receptive field with the same number of parameters. Images taken from [33].

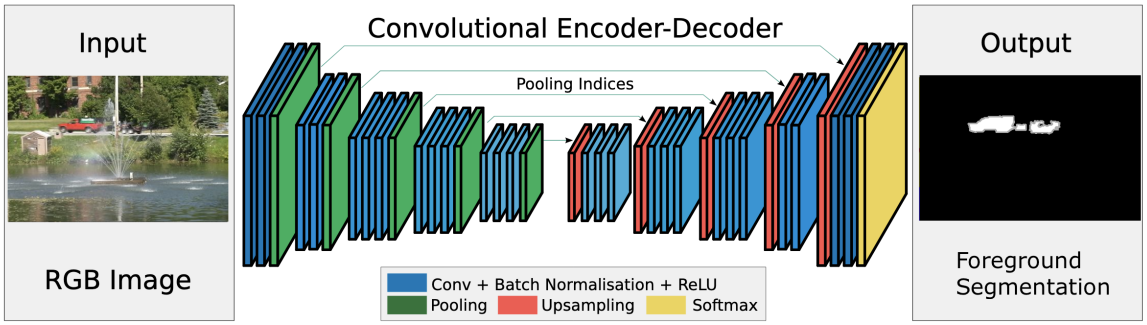


Fig. 2.5 A typical Convolutional Neural Network with an encoder-decoder architecture for background subtraction. The encoder is consisted of multiple Convolution layers which extract semantic features, followed by Pooling layers for dimensionality reduction. The decoder uses Upsampling layers to restore the resolution of the feature maps to its original size, while Convolution layers are used to recover the information lost due to downsampling. The ReLu and Batch Normalisation layers ensure the stability of the training process, while the Softmax layer converts the features of the last layer to pixel-wise probabilities. Parts of this image were taken from [6].

the authors of FCN fuse information from shallow layers that precede pooling layers via feature map concatenation. These skip connections enabled the flow of information to bypass the bottleneck. Although the segmentation result improved significantly, the boundary of the segmented object remained ill-defined.

Ronneberger et al. [103] addressed this issue by adding a decoder, which is symmetric to the encoder with the exception that pooling layers are replaced by transpose convolution layers. Skip connections are employed to concatenate the feature maps of those layers of the encoder that match the resolution of the decoder's layers. Since the decoder has as many layers with as many feature maps as the encoder, it can successfully recover the loss of information caused by the pooling layers. With the aid of the skip connections, the network segmented small neuronal structures with great accuracy. This encoder-decoder architecture, commonly known as *Unet*, is depicted in figure 2.5.

2.4.2 Dilated convolutions

The next breakthrough in image segmentation was achieved with the use of dilated convolutions (Figure 2.4c), which became an essential feature in this domain when Chen et al. [21] achieved state-of-the-art results. In contrast to regular convolutions, dilated (or atrous) convolutions use a kernel that is zero-padded in between its values. The number of zeros inserted is a hyperparameter known as *dilation rate*. Since the kernel is enlarged after this operation, its receptive field is increased. The benefit of dilating the kernel instead of simply using a larger kernel is that the computation cost remains low. Considering that a CNN can easily have hundreds of convolutions, the computation savings can be very significant. In addition, since the field of view of the filters is enlarged, some pooling layers can be safely removed. This means the loss of information in the encoder is limited, and thus the decoder has more details available. Another advantage of dilated convolution is that they can effectively segment objects at multiple scales, when many different dilation rates are used. Prior to dilated convolutions, two main methods were used to alleviate this problem. The first approach dictates to feed the same image at different scales and then aggregate the features [22, 66], either by concatenation or summation. The second approach uses two different streams, one for modelling the local context and another for the global context and then perform feature fusion [104, 102, 63]. However, both techniques can be very expensive in terms of computation, considering the number of additional convolutions. On the other hand, dilated convolutions add no computational overhead. They can either be placed in parallel [21] or sequentially [154, 155], however the parallel structure is considered superior since it helps avoid gridding artifacts efficiently [136].

Zhao et al. [161] extended this approach by adding a pyramid pooling module. The module was appended to the output of the last convolutional layer and consisted of four pooling layers of different kernel sizes and strides. Each pooling layer extracted information

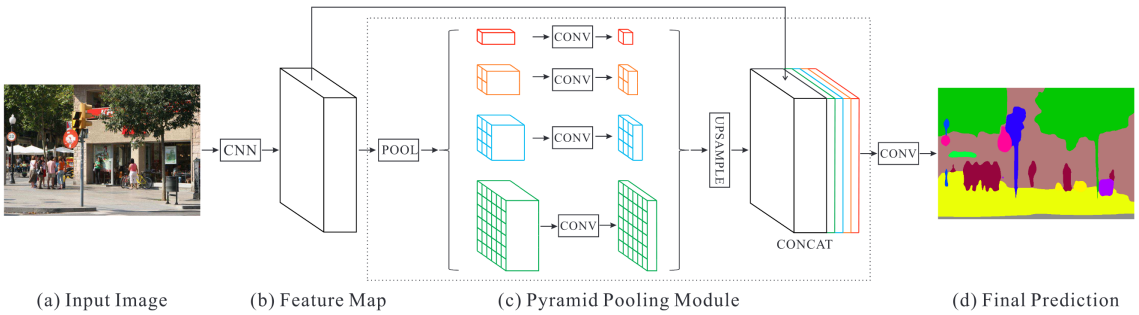


Fig. 2.6 An overview of PSPNet with the groundbreaking pyramid pooling module. The output of the last convolutional layer is fed to the module, which uses pooling layers with kernels of various sizes to extract features from different scales. The output is then upsampled and concatenated. [161]

from different parts of the image in different scales. The output of these layers was then upsampled and concatenated with the output of the previous convolutional layer. Essentially, the objective of the module is to aggregate information taken from different context. With a kernel size equal to that of the original feature map, global context is gathered. Similarly, using smaller kernels, the context becomes local. With this feature, PSPNet became the best model in ImageNet scene parsing challenge 2016, PASCAL VOC 2012 benchmark and Cityscapes benchmark.

2.4.3 Conditional Random Field for boundary refinement

Some methods attempt to refine the segmentation mask using post-processing techniques. The most commonly used method is applying a Conditional Random Field (CRF) on the CNN output. A fully connected CRF is formally defined as follows:

$$E(x) = \sum_i \theta_i(x_i) + \sum_{ij} \theta_{ij}(x_i, x_j), \tag{2.4}$$

where the first and second terms of the equation denote the unary and pairwise potentials respectively. The unary potentials can be replaced by the output of the CNN and only the

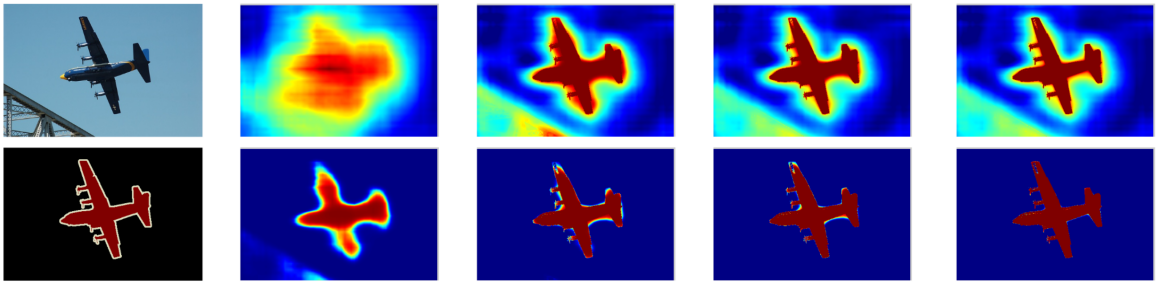


Fig. 2.7 Applying a Conditional Random Field (CRF) to the Deep Convolutional Neural Network (DCNN) output vastly improves the boundaries of the segmentation mask. Image taken from [20].

pairwise potentials are learned. The second term can be broken as

$$(1-\delta_{ij})\sum_{p} w_p k_p(f_i, f_j)$$

, where δ_{ij} is the Kronecker's delta, k is a Gaussian kernel that is weighted by the parameter w and depends on the features f extracted by the pixels i, j. As depicted in figure 2.7, the CRF greatly improved the results.

Various implementations of a CRF were used by a plethora of works [136, 20, 21, 161, 66] as a post-processing step. Others however, attempted to approximate CRF using operations performed by the CNN and therefore integrate it within the network architecture. Zheng et al. [165] reformulated a mean CRF inference as a Recurrent Neural Network (RNN), effectively creating an end-to-end system, trainable by backpropagation. The FCN [82] was used as a backbone network. Liu et al. [81] extended this work by implementing a more efficient system that only required a single iteration of the mean field algorithm of the CRF. Their model was also able to analyze temporal information using a 3D convolutional layer on three consecutive frames.

2.4.4 Multi-task networks

Finally, there has been some research that indicates multi-task CNNs perform better. He et al. [47] implemented a joint image segmentation/object detection system by creating a three-branch CNN with a shared core. The first and second branches were trained for classification (*cls*) and bounding box (*bbox*) prediction respectively, while the third branch performed image segmentation (*iseg*). All three branches were jointly optimised during the training process using a single loss $L = L_{cls} + L_{bbox} + L_{iseg}$. The model was flexible, in the sense that any backbone network can be used. The authors report that the multi-task nature of the network boosts the results of all tasks.

2.5 Background Subtraction with Deep Learning

In this Section we will focus on Deep Learning - based methods for BGS and discuss the most relevant papers. We categorise them based on distinctive characteristics of their architecture, such as input modalities and inherent structures. We also describe new concepts, namely LSTMs and GANs.

2.5.1 Single frame methods

Most BGS methods follow the trend of recent generic image segmentation networks and treat videos as a collection of images while disregarding the temporal information. Since the task of background subtraction can be regarded as binary segmentation (foreground, background), there are many papers that implement ideas from state-of-the-art image segmentation models. For example, Lin et al. [72] use the FCN [82] for background subtraction. The authors concatenate the current frame with the background image channel-wise and feed the 6-channel image to the FCN. Although their model achieved satisfactory results in baseline

videos, it failed when presented with noisy frames or small objects. In addition, a clean background frame is often difficult to obtain in real life scenarios.

More advanced approaches use multi-scale feature aggregation. Following the success of earlier approaches in object detection [17], image segmentation [103, 82], edge detection [143] etc., Zeng and Zhu [156, 157] realise this idea simply by concatenating features from different layers. In their method, a Unet is used with VGG16 as the backbone network. On the other hand, Lim and Keles [71] employ multi-scale inputs, in a similar fashion to studies like [66, 22, 84]. Their model is also reusing weights of VGG16. Wang et al. [141] also adopt the same input preprocessing, however in this case no pre-trained network was used. The authors used a double multi-scale CNN but they devised a cascaded-like architecture, with the output of the first network being fed into the second CNN in order to refine the segmentation result. Although the computational overhead was significant, only minor improvements were obtained. Temporal information was not analyzed in this approach. Zhao et al. [164] also used a similar cascaded architecture, however in this case the first CNN was used for reconstructing the background. The reconstructed image is concatenated with the input and fed to the second CNN for background subtraction. For the first network no pre-trained model was used, whereas the second CNN was based on DeepLab [21]. The two tasks were optimised jointly with a multi-task loss.

Caelles et al. [16] focus on segmenting the primary moving subject on a given video, which holds strong similarities to background subtraction. The authors use a multi-task two-stream FCN with skip paths between the layers and train the first path for contour detection and the second for image segmentation. As with previous models, the VGG architecture [115] was used as the backbone. Although the model was pre-trained on the Imagenet dataset [105], only the first frame of the given video is used for training, which renders their model sensitive to background changes. Post-processing techniques are employed to combine the output of the two streams for the final segmented image. A great limitation of this system

consists of the need to be fine-tuned for each video sequence. In addition, the system will be unable to accurately segment an object that does not appear on the first frame but enters the scene at a later phase. Similarly, Akilan [128] use a three-branch CNN with the first convolutional layer of each branch performing convolutions with kernels of different size, namely 3x3, 5x5 and 9x9. The features were fused at later layers via concatenation. The backbone network that was used was VGG16 [115], with the author reporting a significant boost in performance when the pre-trained weights were used, as opposed to training the network from scratch.

Cinelli et al. [28] implemented a model based on SegNet [6], with the exception that strided convolutions were used for downsampling the feature maps resolution instead of pooling layers. The authors performed experiments with decoders of different types of upsampling methods: bilinear interpolation and single/multi channel deconvolutions with and without activation functions. Unpooling layers were not tested due to the absence of pooling layers in the encoder. A resnet-based decoder with single channel deconvolutions was found the most effective.

Finally, Zeng et al. [32] attempt to improve the output of existing, non - deep learning based models with a CNN. The input of the CNN consists of the output of three background subtraction algorithms, namely SuBSENSE [122], FTSG [137], and CwisarDH [31]. Their model has an encoder-decoder architecture with VGG16 [115] as the backbone model. The main disadvantage of this method is that it is limited by the other algorithms, since if those fail completely, the CNN will be unable to reach satisfactory performance.

2.5.2 Spatio-temporal models

In contrast to the models mentioned thus far, there are some approaches that attempt to model the temporal information as well. This can be accomplished in two ways. The first method

dictates that a small temporal window of n frames is used as input to the model, which then analyses the patterns between adjacent frames by performing 3D convolutions. The stride of the sliding window can be larger than 1, especially in cases where the input video features slow moving objects. In addition, certain methods place the current frame in the center of the window, while others only use past frames.

The second method is through the use of Long Short-Term Memory (LSTM) neural networks. An LSTM network, depicted in figure 2.8, is a recurrent neural network that is specially designed to capture long term patterns. It consists of a cell C_t, which holds the information at the timestamp t, and structures of different functionality, named *gates*. If, given a timestamp t, we denote the input of the LSTM unit as x_t, the output as h_t, the weights and bias of a unit as W and b respectively, and σ as the sigmoid function, then the gates of an LSTM unit can be formally defined as follows:

- the input gate i_t, learns the importance of each element to be stored to the cell, depending on the previous output: $i_t = \sigma(W_i(h_{t-1}, x_t) + b_i)$
- the forget gate f_t, learns which elements are obsolete and need to be removed from the cell: $f_t = \sigma(W_f(h_{t-1}, x_t) + b_f)$
- the output gate o_t, learns what is the optimal output for the given input: $o_t = \sigma(W_o(h_{t-1}, x_t) + b_o)$

Then, the cell can be updated as $C_t = f_t * C_{t-1} + i_t * C_t$, and the output of the unit can be calculated with the following function: $h_t = o_t * tanh(C_t)$, where $tanh$ is the hyperbolic tangent function.

Both methods are useful for modelling temporal information, depending on the task at hand. For example, LSTMs are superior in learning long-term dependencies, since they can store information on their cell for extended periods of time. On the other hand, 3D convolutions require far fewer parameters and can learn spatial and temporal patterns

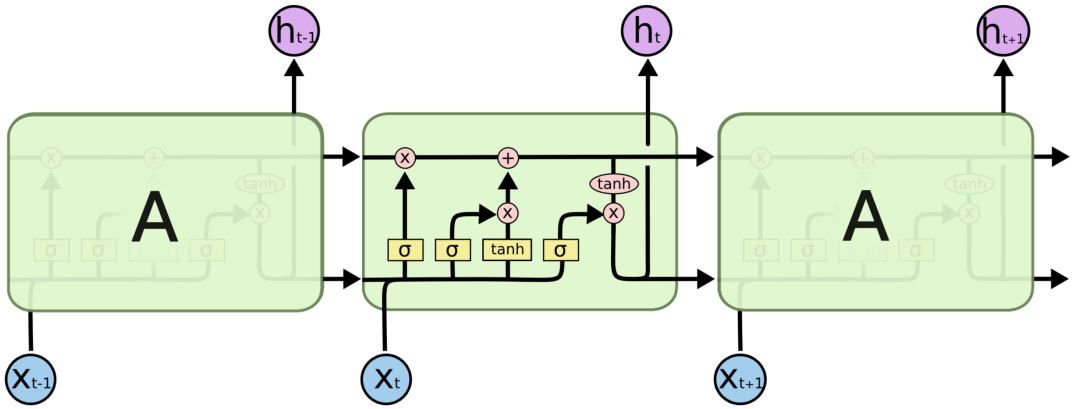


Fig. 2.8 Visualisation of a Long Short-Term Memory unit, a type of recurrent neural network for modelling long term patterns. The symbols $\otimes$ and $\oplus$ denote point-wise multiplication and addition respectively, whereas σ is a sigmoid function. Finally, x_i and h_i are the unit's input/output. Image taken from [90].

simultaneously. In addition, the temporal window can be adjusted by changing the stride of the input frames sequence. Below, we review papers that use either method, or both.

2D models

Chen et al. [25] use an encoder-decoder architecture with various backbone networks - VGG16 [115], ResNet [48] and GoogLeNet [126]. The output of the FCN is fed to an LSTM for temporal modelling. After, a Spatial Transformer Network [54] is appended to learn rotation invariant features, which is can improve results when camera motion is present. Finally, the result was refined with a CRF, which was implemented as a recurrent neural network as in [165]. An attention module was embedded to fuse the outputs of the FCN and the LSTM. Although the model performed well generally, it received poor results in videos with low framerate and camera movement.

3D models

To the best of our knowledge, we were the first to use a 3D CNN for the task of background subtraction [109]. Gao et al. [39] used a small 3D CNN with only two convolutional layers and a fully connected layer. Their network operated in a patch-based manner, therefore missing the global context of the input image. Hu et al. [51] use a two-branch 3D CNN with an encoder-decoder architecture that has a temporal window of 12 frames, the current frame being at the center. While the encoder operated in 3D, the decoder was 2D. Each branch performed in different temporal scale, while dilated convolutions were used for spatial multi-scale feature extraction. The authors employed a slow temporal feature fusion, in a similar fashion to [109]. To capture long-term temporal information, a two-unit Long Short-Term Memory (LSTM) neural network was used. Finally, the network was trained with weighted focal loss. Wang et al. [142] used a very similar two-branch 3D architecture, although the input of their network was 15 consecutive previous frames plus the current frame. Also, they followed a sequential feature fusion style with no LSTM units. Their CNN was trained in three stages: first, the encoder was trained in a large dataset for the task of action recognition, then the decoder was trained for background subtraction, and finally the decoder was fine-tuned for each specific video. Finally, Akilan et al. [1] implement a 3D CNN-LSTM model that differs to the traditional encoder-decoder structure. Instead, the network consists of mini blocks of strided convolutions followed by upsampling layers realised by 3D transpose convolution. The output of each block is concatenated with the output of the previous block. There are also longer skip connections between the blocks. Two LSTM modules are used, the first being placed in the middle of the network, while the second is appended to the output of the penultimate convolutional layer. In contrast to previous approaches which use a 2D decoder, this model only employs 3D convolutions.

2.5.3 Patch-based methods

Some background approaches are patch-based, meaning that these models do not accept the whole frame as input, but only regions. Therefore, for the segmentation of a given image, several passes need to be run. Braham et al. [14] followed a patch-based Deep Learning approach, using a CNN with the background and the foreground as inputs. The background was constructed by computing the temporal median value of 150 frames of the video sequence but was not being updated, making the model prone to misclassification in highly dynamic backgrounds. Their model was scene-specific, meaning that it needed to be retrained for each video sequence. This limitation was addressed by Babaee et al. [5], who used a 3-level system for background generation including a motion detector. The images were then fed into a CNN in a patch-based manner, and the output was post-processed using a spatial-median filter.

Liu et al [79] proposed a method based on sparse signal recovery which exploited group property information in both spatial and temporal domains. Javed et al. [55] improved [34] by incorporating spatio-temporal constraints and reported better performance.

2.5.4 GAN-based models

Here we discuss methods which perform background subtraction using a Generative Adversarial Network (GAN). Originally proposed by Goodfellow et al. [40], GANs are deep learning models that are comprised of two distinct networks: the Generator (G) and the Discriminator (D). Let $I = \{i_1, i_2, \ldots, i_n\}$ be the domain of the training data and $z \in \mathbb{R}$ be variables of random noise. The generator receives a vector z of fixed length as input, initialised by noise. Through a series of deconvolutions, it upsamples z until it reaches the wanted size. The task of the generator is to produce new samples $G(z)$ that match the distribution of the input images i as much as possible. On the other hand, the discriminator accepts a training

image, i, or an image created by the generator, $G(i)$, in an alternating fashion. Its task is to classify the given input as either real of fake. While D is trained to maximise the probability $p = D(x)$ of making correct classifications, G is trained to minimise the probability of D correctly classifying $G(z)$ as fake. Formally, the two networks are jointly trained with a minimax loss:

$$L_{GAN} = arg\,min_G\,max_D\,f(G,D) = E(log(D(i)) + E(log(1 - D(G(z)))), \quad i \in I,\ z \in \mathbb{R}. \tag{2.5}$$

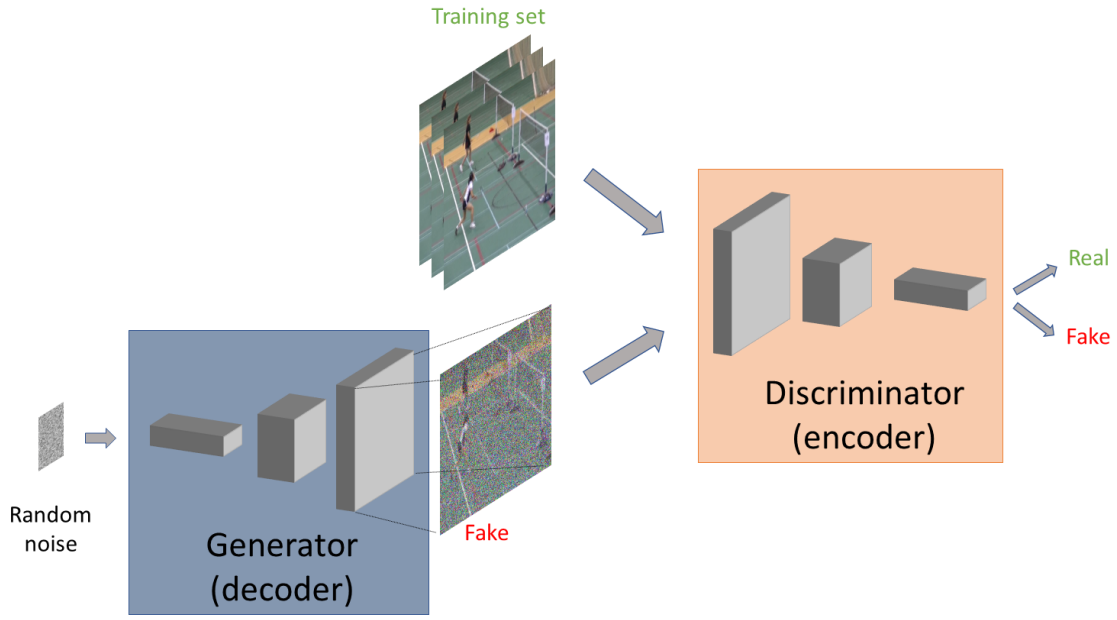
Since the goal is to achieve balance between the generator's and discriminator's tasks, special care needs to be taken so that either loss does not saturate. This can be controlled by updating the weights of D more often than those of G [40]. Some approaches incorporate a similarity loss L_S that enforces $G(z)$ to be as close to the real image as possible, besides fooling the discriminator [94]. Commonly used similarity losses include the L1 and L2 norms [53]: $L_1 = E(||i - G(z)||_1)$ (similarly for the L_2 norm). In such cases, the loss becomes:

$$L_G = arg\,min_G\,max_D\,f(G,D) = L_{GAN}(G,D) + \lambda L_S(G), \tag{2.6}$$

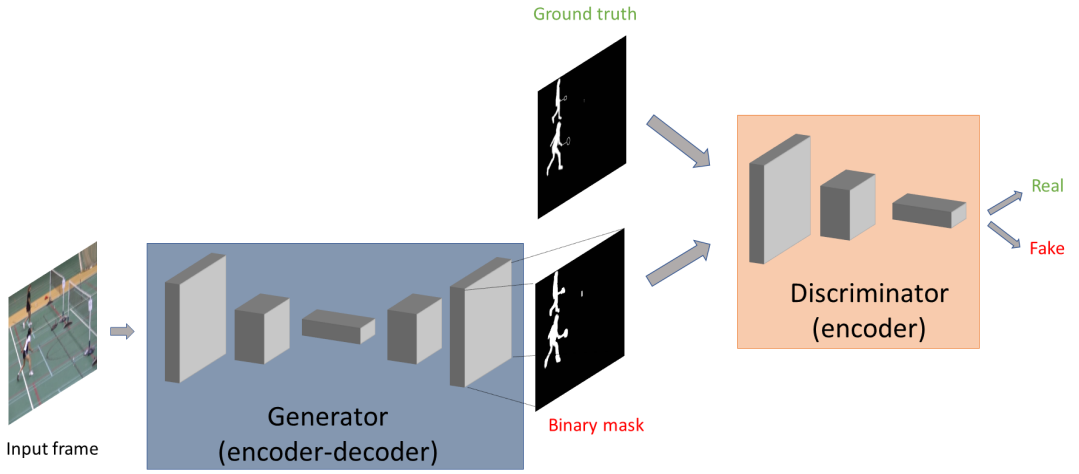
where λ is a scaling parameter.

A depiction of this system is given in figure 2.9. These networks have been proven to be very successful in generic image-to-image translation [168, 152, 53], though it can also be used for background subtraction. However, many papers have shown that GANs are capable of modelling the illumination of an image exceptionally well [4, 138, 74]. Thus, in our work [107] we have used GANs for illumination-aware background subtraction.

Other works use GANs directly for background subtraction. Bakkay et al. [7] implemented BScGAN, a conditional GAN for background subtraction. Instead of noise, the generator accepts as input the current frame concatenated with the background image of



(a)



(b)

Fig. 2.9 Illustration of a Generative Adversarial Network for (a) image generation and (b) background subtraction. In the case of creating fake images, the generator is a decoder CNN that accepts random noise as input and though a series of convolutions and upsampling layers produces a fake image. For the task of background subtraction, the generator becomes an encoder-decoder CNN that accepts the input frame as input and produces the binary segmentation mask. The discriminator is a decoder in both cases and its task is to classify the input as fake or ground truth image.

a given video sequence, and produces the binary segmentation maps. Since its input is a full-sized image, an encoder-decoder architecture is used. Except for the first and last convolutional layers of the encoder, the rest are residual blocks initialised from Resnet101 [48]. The decoder employs transpose convolutions for upsampling the deep feature maps. On the other hand, the discriminator consists of four convolutional layers of stride 2, for downsampling the resolution of the feature maps. A fully connected layer with a sigmoid activation function outputs the classification probability. The equation 2.6 is used as the loss function, with L_2 norm being the similarity loss and $\lambda = 10$. Although the model performed well in most categories, having a clean background frame is often an unrealistic assumption.

Recently, Patil and Murala [95] implemented a triple-step method for background subtraction with GANs. First, a GAN is used to generate the background image, when given N frames of a video. The discriminator classifies the given image as either the generator's output or as the ground truth background image. The second step of the approach involves estimating the motion saliency map by simple image subtraction between the current frame and the generator's output. Finally, another GAN is used to generate the binary segmentation masks, given the motion saliency map. The reported results of this approach were good, however there are some issues. First, similarly to the previous GAN-based method, the background image is needed to train the model. Secondly, the complicated three-step pipeline not only is cumbersome, but also has a high risk of failure. That is because the foreground segmentation GAN depends on the first GAN producing an accurate background image; if this step fails, the second GAN will be unable to recover. Finally, the model was not tested in videos with varying illumination that could challenge the background model.

2.6 Conclusion

In this chapter we have reviewed the most relevant and influencial papers in image segmentation and background subtraction. In addition, we have introduced the basic concepts that are used in this thesis. Although early approaches can be effective in tackling some of the challenges of this research area, deep learning algorithms consistently outperform these by a large margin. That is because very deep neural networks can approximate extremely complex functions and are able to extract semantic information from the input. Furthermore, they can model both spatial and temporal information with either 3D convolutions or LSTMs. Another advantage of deep learning models is that they are usually end-to-end systems, although pre-processing and post-processing techniques can be used to boost the performance.

Even though existing methods address certain challenges regarding background subtraction, there are some issues which remain unsolved. Wang et al. [141] were the first to employ deep CNNs for the task of background subtraction, however the temporal information was not considered. In addition, Vosters et al. [135] implemented a method robust to illumination changes, however their approach was not deep learning based and thus, no semantic information were considered. In this thesis we address these limitations and present end-to-end models which outperform the state-of-the-art.

Gaussian Mixture Models (Unsupervised)	
Method	Description
Friedman [38]	Three gaussian mixture model
Zivkovic [170]	GMM of adaptively estimated number of gaussians
Siva [117]	GMM + Probabilistic model
Boulmerka [12]	GMM + Statistical model
Akilan [2]	GMM + multi-feature model
Chen [23]	GMM with hierarchical superpixel segmentation and optical flow
Shen [113]	GMM in low dimensional data
Pilet [98]	Illumination-based GMM
Tuzel [132]	3D GMM
Kernel Density Estimates (Unsupervised)	
Elgammal et al. [36]	KDE with spatio-temporal constraints
Zivkovic [171]	Balloon estimator - based KDE with decay
Mittal [89]	Hybrid KDE with optical flow
Zhu et al. [169]	KDE with local post-processing
Berjón [10]	Temporally updated KDE with bayesian classifier
Principal Component Analysis (Unsupervised)	
Oliver [91]	PCA-based background model
Robust PCA [130]	RPCA via Principal Component Pursuit (PCP)
Liu [78]	Efficient PCP - running in parallel
Candès et al. [18]	Efficient RPCA with sparse matrices
Zhou [166]	Hybrid KDE with optical flow
Mittal [89]	Bilateral Random Projections - based PCP for noisy data
Zhang [160]	Covariance matrix - based BGS with Local Binary Patterns
Ibadi [34]	Tree-structured RPCA
Xin [145]	RPCA with generalized fused lasso regularization
Vosters [134]	PCA-based BGS with statistical illumination model
Zhang [160]	Covariance matrix - based BGS with Local Binary Patterns

Table 2.1 Summary of background subtraction/foreground segmentation methods

Deep Learning Models (Supervised)			
Method	Dataset	FM	Description
Lin [72]	CDnet2014	0.6874	FCN with current and background frames as input
Zeng [156, 157]	CDnet2014	0.9870	VGG-based Unet with skip connections
Lim [71]	CDnet2014	0.9804	VGG-based Unet with multi-scale inputs
Wang [141]	CDnet2014 SBI2015	0.95 0.8932	Cascaded FCN architecture
Zhao [164]	CDnet2014	0.8124	Cascaded FCN with background model
Akilan [128]	CDnet2014	0.8605	Three-branch FCN
Cinelli [28]	CDnet2014	0.849	SegNet-based architecture
Zeng [32]	CDnet2014	0.8243	VGG as a refinement model
Chen et al. [25]	CDnet2014 LASIESTA [30]	0.8772 0.9042[1]	FCN + LSTM + STN + CRF
Gao [39]	Mixed	0.9510	Tiny patch-based CNN
Hu [51]	CDnet2014	0.9615	3D FCN + LSTM
Wang [142]	CDnet2014	0.9620	Dilated 3D FCN
Akilan [1]	CDnet2014	0.9574	3D Encoder-Decoder CNN-LSTM
Braham [14]	CDnet2014	0.9046	Scene-specific patch-based CNN
Babaee [5]	CDnet2014 Wallflower	0.7548 0.7512	3-step BGS with BG model, patch-based CNN and post-processing
Bakkay [7]	CDnet2014 BMC	0.9763 0.945	Conditional GAN with ResNet backbone and clean BG frame input
Patil [95]	CDnet2014	0.9697	3-step BGS with two cascaded GANs for BG generation and saliency detection

Table 2.2 Summary of Deep Learning - based background subtraction/foreground segmentation methods

Chapter 3

End-to-End Video Foreground Segmentation with 3D Convolutional Neural Networks

In this chapter, we follow the success of Deep Learning in Computer Vision and present an end-to-end system for foreground segmentation in videos. Our model is able to track temporal changes in a video sequence by applying 3D convolutions to the most recent frames of the video. Thus, no background model is needed to be retained and updated. In addition, it can handle multiple scenes without further fine-tuning on each scene individually. We evaluate our system on the largest dataset for change detection, CDnet [140], with over 50 videos which span across 11 categories. Further evaluation is performed in the ESI dataset [135] which features extreme and sudden illumination changes. Our model surpasses the state-of-the-art on both datasets according to the average ranking of the models over a wide range of metrics.

Portions of this chapter have been previously published in [109].

3.1 Introduction

The proposed deep learning model aims to improve frame-by-frame continuity and reduce noise by modelling the input data not only spatially but also temporally. In contrast to existing methods, our system is entirely end-to-end with no need for background model, preprocessing or post-processing methods. Unlike other CNN-based methods, our approach is not scene specific and can analyze a variety of scenes with no further training. In addition, it is able to track background changes of all kinds, analyse and fuse these into the segmentation process. We adopt a multi-scale approach that allows the network to incorporate features from shallow, mid-level and deep layers. As a result, our model has access to rich spatial and temporal information. The superiority of our method is demonstrated by experiments in two different datasets, in both of which we surpass the state-of-the-art. In particular, full evaluation is performed on the CDnet dataset [41, 140], one of the most complete datasets in the domain of background subtraction with over 50 video sequences spanning 11 categories. For further validation of the superiority of our model in extremely challenging environment conditions, we use the ESI dataset [135]. This dataset features rapidly changing illumination conditions and we outperform the authors' method significantly. Inspired by Ji et al. [58] and Karpathy et al. [63], we use 3D Convolutional Neural Networks for achieving an end-to-end approach for foreground segmentation. The rest of this chapter is structured as follows: Section 3.2 is devoted to the analysis of our methodology and network architecture. In Section 3.3 we describe the datasets and the evaluation metrics used in this study and we present our results. Finally, a conclusion is provided in Section 3.4 and future work is discussed.

3.2 Methodology

The proposed method is carefully designed to encompass both spatial and temporal information. This is of paramount importance for the task of video segmentation, as the model must be able to:

1. Track recent changes in the background that might happen either naturally or artificially.

2. Track movement of the foreground objects.

To this end, instead of operating in a frame by frame basis, we harvest information from the 10 most recent frames for the segmentation of each frame. This can be done via 3D convolutions across all frames, a method which is able to satisfy both points mentioned above.

Formally, a 2D convolution is defined as follows:

$$Conv_{2D}(m,n) = \sum_{k=0}^{K-1} \sum_{l=0}^{K-1} W(k,l) \otimes x(m+k, n+l), \tag{3.1}$$

where $\otimes$ denotes the convolution operation, W a kernel of size K, x the input image/feature map, m,n the index of the first pixel of the input, k,l the index of the element of the kernel. The final output of a convolutional layer adds a bias b to the result of the convolution operation and applies a non-linear activation function σ like the hyperbolic tangent or a rectified linear unit (ReLu). As a result, it is deduced that the dimensionality of the output volume depends on the sliding window stride s, kernel size K, zero-padding p and input image size I and is given by $\frac{I-K+2p}{s} + 1$.

For the case of 3D convolutions, a 3D kernel must be used instead. Therefore, the above formula will be extended towards the temporal (depth) dimension and is formally defined as follows:

$$Conv_{3D}(q,m,n) = \sum_{t=0}^{T-1} \sum_{k=0}^{K-1} \sum_{l=0}^{K-1} W(t,k,l) \otimes x(q+t, m+k, n+l), \tag{3.2}$$

where T stands for the length of the input sequence, q is the index of the first pixel of the input image/patch along the third dimension, and t is the index of the element of the kernel in the third dimension. Here, the 3D kernel moves in all three directions of the data: height, width and depth, and therefore learning patterns which are formed in all dimensions.

3.2.1 Network architecture

Temporal connectivity The full network architecture consists of 6 groups of convolutional layers and is depicted in figure 3.1. Since the vast majority of the images in the dataset have a resolution of 320x240 pixels, we rescale the remaining images to this size by nearest-neighbor interpolation and keep a fixed resolution. The input window is consisted of 10 consecutive frames. Instead of simply concatenating all frames altogether and feeding it to a single convolutional layer, we follow a different approach. Inspired by Karpathy et al. [63], we adopt a slow fusion of the shallow features. More specifically, the input is divided into groups of 4 frames with stride 2 and connected to 4 respective convolutional layers. Thus, these layers capture not only spatial but also motion information. With pooling layers in-between, the next groups of 3D convolutions are slowly merged until they conclude to a layer with no chronologically parallel convolutions. With the dimension of time now being 1, we continue with 2D convolutions. After each convolution layer, we apply the max function $f(x) = \max(0,x)$ in the form of ReLu layers.

This architecture has the advantage of eliminating noisy input frames. In the event of a frame being affected by any kind of noise, caused by varying illumination, mechanical failure etc. the model still has access to better quality information from the rest of the input

frames. Thus, the binary mask output is ensured to be more robust compared to single-frame models.

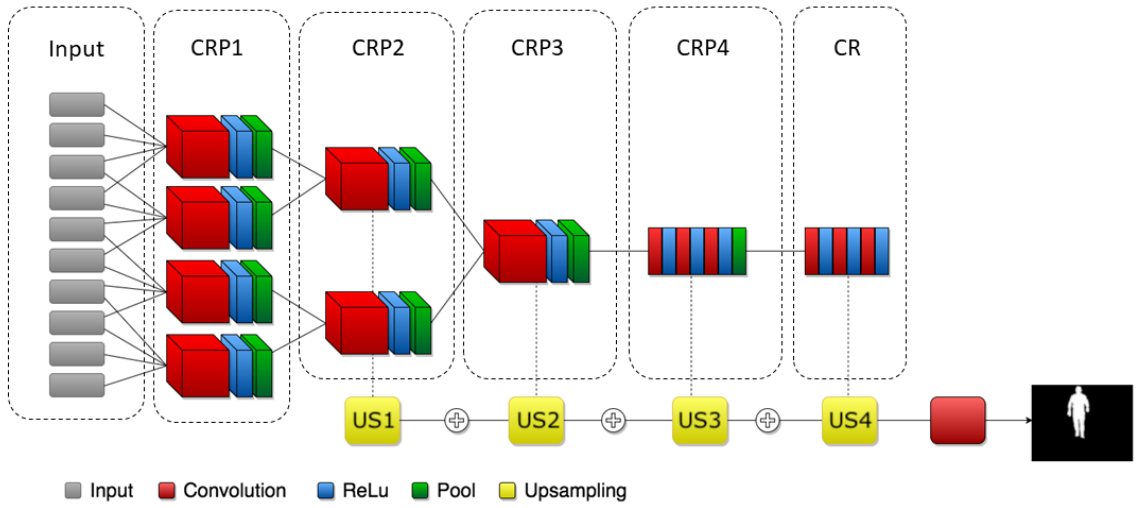


Fig. 3.1 The network architecture. The input comprises a video of 10 frames which is connected to the first group of layers, CRP1, in groups of 4 frames with stride 2. CRP1 is then connected to CRP2 in the same manner and CRP3 has access to the features of all frames. CRP4 is performing 2D operations only, while CR has no pooling layer. The upsampling layers US1, US2, US3 and US4 are connected to CRP2, CRP3, CRP4 and CR respectively and are concatenated before applying the final convolution. The full layer specifications are presented in table 3.2. It should be noted that cubes indicate 3D operations across the temporal dimension, while rectangles indicate 2D (spatial only) operations. The plus sign indicates concatenation.

Multi-kernel upsampling In order to retain fine-detail information from the input images, we perform upsampling by 2D bilinear interpolation. This is realised by 2D transpose convolutions using kernels that are initialised for bilinear interpolation and remain fixed during training. To control the upsampling rate, we change the kernel size and stride. Each kernel is used on layers of different convolutional groups so that both low-level and high-level information is utilized. To upsample feature maps from layers CRP1 and CRP2, which are 3D, we first project them to 2D by performing 3D convolutions with kernels of depth equal to the depth of the input feature maps. The upsampled feature maps are then concatenated

and connected to a final convolutional layer that produces the final prediction map. More specifically, four kernels are used with their respective kernel size (K) and stride (S) being (K, S) = (4, 2), (8, 4), (16, 8) and (32, 16). Full details of our architecture are listed in table 3.2.

3.2.2 End-to-end training

As mentioned above, we adopt no preprocessing techniques on the input data. Moreover, we train on each dataset as a whole, without the need of fine-tuning each video sequence separately. In addition, all layers are trained jointly.

In regard to the creation of the train/test set, each video V is first grouped into short video sequences s of 10 frames in an overlapping manner:

$$V = \{s_1, s_2, ..., s_n\},$$

where n the total number of frames and $s_i = \{f_i, f_{i+1}, \ldots, f_{i+10}\}, i \in \{1, \ldots, n-10\}$.

Then, for each video sequence with N number of frames, we select a random integer $i \in 0, ..., 0.7 \times N - 10$ which represents the starting index of the testing set. Its length is 30% of the total frames of the video. Of the remaining frames, we remove those which overlap with the test set and the rest are used for training. To balance the two classes of the dataset, we only keep 20% of the images with no foreground. For the ESI dataset, 100 continuous frames of each video sequence are reserved for testing and the rest are used for training, with an exception of the short clip of *Scene2* where only 76 frames were used for testing. The supervision is consisted of the annotated binary segmentation of the most recent frame. We train on the CDnet dataset [140] for 500,000 iterations using a starting learning rate of 10^{-8} and gradually decreased. We use a Stochastic Gradient Descent (GSD) solver and a high momentum of 0.9. For the loss function, we use the pixel-wise cross entropy-loss, which is defined as:

$$E = \frac{-1}{n} \sum_{n=1}^{N} [p_n \log \hat{p}_n + (1 - p_n) \log(1 - \hat{p}_n)] \tag{3.3}$$

where N denotes the number of pixels, $p_n \in \{0, 1\}$ is the label of the pixel and $\hat{p_n} \in [0, 1]$ is the probability prediction, obtained by applying the sigmoid function $f(x) = (1 + \exp(-x))^{-1}$ to the prediction heatmap of the last convolutional layer.

On the ESI dataset [135], we fine-tune for 20,000 iterations using the weights of the previous network. All parameters remain unchanged with the exception of the base learning rate, which is adjusted to 5×10^{-9}. Since this dataset is significantly smaller than CDnet with less than 3,000 annotated frames, we use dropout layers to avoid overfitting. Specifically, a dropout layer with a ratio of $dr_1 = 0.2$ is appended to the third convolutional block after the final 3D convolution. An additional dropout layer with a ratio of $dr_2 = 0.3$ is incorporated into the fourth convolutional block and a final layer with $dr_3 = 0.5$ is appended to the final convolutional layer of the fifth block. We also extend the dataset by including 2,450 non-annotated frames with no foreground in our training set.

3.2.3 Implementation

For the development and evaluation of our model we use Caffe [59]. To perform 3D convolutions and 3D pooling, we use the C3D branch [131]. The computer used for experiments is a 64GB RAM machine with Intel Core i7-5960X CPU @ 3.00GHz x 16 and 4 GeForce GTX TITAN X Graphics Processing Units.

Computational time Training of the full model is completed within 20-24 hours for the CDnet2014 dataset and 6-8 hours for ESI.

3.3 Evaluation

As mentioned above we use two different datasets for evaluating our model. This section reports the results of our experiments and is organized as follows: First, we present the datasets that have been selected for the evaluation of our model and a full account on the reason for the selection is provided. Next, we introduce the evaluation metrics and list their formulas. Finally, we demonstrate our results using a plethora of tables, graphs and plots and discuss how we compare against the state-of-the-art.

3.3.1 Datasets

Finding the proper dataset for the evaluation of a model is often a cumbersome task. There are certain requirements to be satisfied, especially in the field of Deep Learning. First and foremost, a large amount of annotated images is an absolute necessity for the convergence of the model. Secondly, the video sequences of the dataset must cover a wide variety of categories to provide an exhaustive testing of the method's capabilities. Lastly, a good dataset must be acknowledged by the research community and used in a plethora of publications; not only for boosting its credibility, but also for the direct comparison between the models.

CDnet dataset The ChangeDetection.net dataset [41, 140] is the only dataset that addresses all three points mentioned above in full:

1. Sufficient annotated data: Over 150,000 annotated frames for training, validation and testing.

2. A multitude of categories: The majority of the background subtraction challenges are covered by the wide range of video categories, namely the following: Bad Weather, Low Framerate, Night Videos, Pan-Tilt-Zoom (PTZ), Turbulence, Baseline, Dynamic Background, Camera Jitter, Intermittent Object Motion, Shadow, and Thermal.

3. Benchmarking: The dataset has been used for the evaluation of numerous background subtraction algorithms over the years. The majority of them -39 in total- are extensively evaluated on seven different metrics and their results are publicly hosted on the website.

Therefore, we use CDnet as our primary dataset.

ESI dataset We select the ESI dataset [135] for the evaluation of our model under rapidly changing illumination conditions, a category that is missing from the CDnet dataset. There are 5 video sequences in total (Scene1, Scene2, Chair, Sofa and Walking) filmed in three unique backgrounds, which feature sudden lighting changes caused by various means, such as different light sources being turned on and off as well as allowing and blocking sunlight into the room. These changes occur in a matter of seconds and as a result are significantly more challenging than gradual lighting changes caused by the transition of the day into night. As a result, the ESI dataset is a valuable extension of the evaluation process.

3.3.2 Metrics

All metrics used in this Chapter are based on the numbers of correctly/incorrectly classified pixels, which can be defined for a given image as follows:

- True Positive (TP): The number of correctly classified foreground pixels.
- True Negative (TN): The number of correctly classified background pixels.
- False Positive (FP): The number of incorrectly classified background pixels.
- False Negative (FN): The number of incorrectly classified foreground pixels.

We utilize the entirety of the seven metrics which are used to evaluate the models published on the CDnet website. Namely, the following: *Recall* or *True Positive Rate, Specificity, False*

Positive Rate (FPR), False Negative Rate (FNR), Percentage of Wrong Classifications (PWC), F-Measure (FM) and *Precision*. The formulas of these metrics are given below [140].

$$Recall = \frac{TP}{TP+FN} \tag{3.4}$$

$$Specificity = \frac{TN}{TN+FP} \tag{3.5}$$

$$Precision = \frac{TP}{TP+FP} \tag{3.6}$$

$$FPR = \frac{FP}{FP+TN} \tag{3.7}$$

$$FNR = \frac{FN}{TP+FN} \tag{3.8}$$

$$PWC = \frac{FN+FP}{TP+FN+FP+TN} \times 100 \tag{3.9}$$

$$FM = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3.10}$$

For further evaluation, we also use Receiver Operating Characteristic (ROC) curves which demonstrate the TPR-FPR tradeoff as the segmentation threshold is altered. Finally, qualitative evaluation is provided in the form of image comparison between our model output versus the state-of-the-art and the ground truth.

3.3.3 Results

CDnet 2014

Due to space constraints, we only provide full metric comparison among the 10 top models of this dataset as published online [42]. More specifically, the following state-of-the-art algorithms are considered: Cascade-CNN [141], IUTIS-5 [11], IUTIS-3 [11], DeepBS [5], PAWCS [121], SuBSENSE [122], WeSamBE [60], SharedModel [24], FTSG [137] and

M4CD Version 2.0 [3]. As depicted in table 3.1, our model surpasses all other algorithms in every metric. With the exception of Cascade-CNN, the rest of the models achieve significantly lower accuracy.

Table 3.1 Overall statistics across all categories on CDnet2014 for the top 10 models [1]

Model	Recall	Specificity	FPR	FNR	PWC	F-Measure	Precision
Ours	0.9609	0.9984	0.0016	0.0391	0.265	0.9507	0.9499
Cascade CNN [141]	0.9506	0.9968	0.0032	0.0494	0.4052	0.9209	0.8997
IUTIS-5 [11]	0.7849	0.9948	0.0052	0.2151	1.1986	0.7717	0.8087
IUTIS-3 [11]	0.7779	0.9940	0.0060	0.2221	1.2985	0.7551	0.7875
DeepBS [5]	0.7545	0.9905	0.0095	0.2455	1.9920	0.7458	0.8332
PAWCS [121]	0.7718	0.9949	0.0051	0.2282	1.1992	0.7403	0.7857
SuBSENSE [122]	0.8124	0.9904	0.0096	0.1876	1.6780	0.7408	0.7509
WeSamBE [60]	0.7955	0.9924	0.0076	0.2045	1.5105	0.7446	0.7679
SharedModel [24]	0.8098	0.9912	0.0088	0.1902	1.4996	0.7474	0.7503
FTSG [137]	0.7657	0.9922	0.0078	0.2343	1.3763	0.7283	0.7696
M4CD Version 2.0 [3]	0.7885	0.9841	0.0159	0.2115	2.3011	0.7038	0.7423

[1] Green color indicates first place, with blue and red showing second and third place respectively. The results of the state-of-the-art are taken from the CDnet website [42].

Table 3.2 In-depth look of our network architecture. Each row represents the full specifications of each group of layers, as depicted vertically from left to right in figure 3.1. CRP: Convolutional-ReLu-Pool, CR: Convolutional-Pool, US: Up-Sampling, FC: Final-Convolution

Layer	Dimensionality	Kernel Size	Stride	Pad	Channels	Pooling Size	Pooling Type
CRP1	3D	3x3x4	1	1	64	2x2x1	MAX
CRP2	3D	3x3x2	1	1	128	2x2x1	MAX
CRP3	3D	3x3x16	1	1	256	2x2x1	MAX
CRP4	2D	3x3	1	1	512	2x2	MAX
CR	2D	3x3	1	1	512	-	-
US1	2D	4x4	2	1	16	-	-
US2	2D	8x8	4	1	16	-	-
US3	2D	16x16	8	1	16	-	-
US4	2D	32x32	16	1	16	-	-
FC	2D	1x1	-	-	1	-	-

Table 3.3 Results on unseen videos

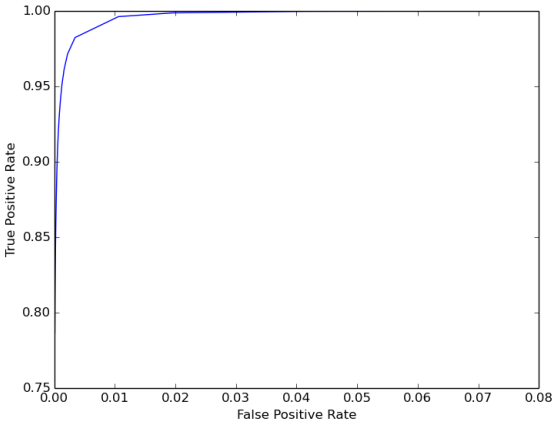
Category	Train set	Test set	Recall	Precision	F-Measure
Baseline	Office Pedestrians	Highway PETS2006	0.3	0.1385	0.1896
Turbulence	Turbulence0 Turbulence1	Turbulence2 Turbulence3	0.09	0.5347	0.1562
Shadow	Backdoor Bungalows Cubicle	Bus Station People In Shade Copy Machine	0.6609	0.7924	0.7207
Dynamic Background	Canoe Fountain01 Fall	Boats Fountain02 Overpass	0.0557	0.0182	0.0275
Thermal	Corridor Dining Room Lake Side	Library Park	0.8992	0.8767	0.8878
Low Framerate	Port Turnpike	Tram Crossroad Tunnel Exit	0.165	0.7185	0.2684
Camera Jitter	Badminton Boulevard	Sidewalk Traffic	0.6805	0.171	0.2733
PTZ	Intermittent Pan Zoom In Zoom Out	Two Position PTZ Cam Continuous Pan	0.2181	0.4868	0.3012
Night Videos	Bridge Entry Fluid Highway Busy Boulevard	Street Corner At Night Tram Station Winter Street	0.4289	0.1195	0.187
Intermittent Object Motion	Abandoned Box Winter Driveway Parking	Street Light Sofa Tram Stop	0.6313	0.0609	0.111
Bad Weather	Blizzard Skating	Snowfall Wet Snow	0.7827	0.4738	0.5577

Table 3.4 Category-wise results on CDnet dataset

Categories	Recall	Specificity	FPR	FNR	PWC
Bad Weather	0.9621 ± 0.0122	0.9981 ± 0.0002	0.0011 ± 0.00025	0.0311 ± 0.00301	0.185 ± 0.0085
Low Framerate	0.9348 ± 0.0176	0.9975 ± 0.00024	0.0039 ± 0.00018	0.0756 ± 0.00678	0.4981 ± 0.0287
Night Videos	0.9137 ± 0.0166	0.9928 ± 0.00031	0.0071 ± 0.00013	0.0825 ± 0.00724	0.6291 ± 0.01928
PTZ	0.8854 ± 0.0164	0.9991 ± 0.00014	0.0008 ± 0.00004	0.1123 ± 0.0121	0.225 ± 0.0173
Turbulence	0.9013 ± 0.0198	0.9994 ± 0.00009	0.0007 ± 0.00006	0.0978 ± 0.00825	0.079 ± 0.00512
Baseline	0.9749 ± 0.0132	0.9993 ± 0.00008	0.0006 ± 0.00004	0.024 ± 0.0174	0.1521 ± 0.00879
Dynamic BG	0.9651 ± 0.0089	0.9988 ± 0.00018	0.0008 ± 0.00024	0.0343 ± 0.00224	0.1578 ± 0.00199
Camera Jitter	0.9751 ± 0.0116	0.9974 ± 0.00025	0.0024 ± 0.00014	0.0371 ± 0.00287	0.3976 ± 0.0205
Intermittent OM	0.9516 ± 0.0172	0.9979 ± 0.00027	0.0012 ± 0.00015	0.0425 ± 0.00375	0.4571 ± 0.00373
Shadow	0.982 ± 0.0125	0.9984 ± 0.00028	0.0027 ± 0.00016	0.0228 ± 0.00211	0.2391 ± 0.00831
Thermal	0.9848 ± 0.0064	0.9995 ± 0.00011	0.0005 ± 0.00004	0.0129 ± 0.00108	0.0912 ± 0.00197

Categories	F-Measure	Precision
Bad Weather	0.9525 ± 0.0039	0.9482 ± 0.0148
Low Framerate	0.8895 ± 0.0052	0.8656 ± 0.015
Night Videos	0.8528 ± 0.0068	0.8393 ± 0.0178
PTZ	0.908 ± 0.0035	0.9478 ± 0.011
Turbulence	0.8869 ± 0.0052	0.8845 ± 0.0125
Baseline	0.9753 ± 0.0025	0.9754 ± 0.0089
Dynamic BG	0.9587 ± 0.0046	0.9503 ± 0.0142
Camera Jitter	0.934 ± 0.0037	0.9278 ± 0.0124
Intermittent OM	0.9711 ± 0.0017	0.9921 ± 0.0113
Shadow	0.9711 ± 0.003	0.9698 ± 0.0198
Thermal	0.989 ± 0.0023	0.9689 ± 0.0122

Fig. 3.2 ROC curve on CDnet dataset



We also present the full results of our model in every category in table 3.4. In general, our model achieves excellent scores in all metrics. Even in categories that either contain a lot of noise like Bad Weather or challenge the background model such as Dynamic Background and Shadow, our results remain outstanding. However, there are some categories in which our results are slightly lower. In the Night Videos category our model presents a higher number of false positives in general, mainly because of huge reflections of the headlights of the cars that occur in the videos, in respect to the small total size of foreground. Likewise in the category of Turbulence, the small size of the foreground objects in conjunction with the noise in the image leads to low false positives on the one hand, but at the expense of an increase in the false negatives. There is also a higher than average false positives in the categories of Low Framerate and PTZ, caused by the large inconsistency between the frames. This conflicts with the assumption of continuity between consecutive frames that is inherent in our architecture. Figure 3.2 depicts the ROC curve of our evaluation, which was generated using 20 threshold values within $(1 \times 10^{-5}, 0.8)$. It is clearly shown that our model accuracy is excellent, and the total area under the curve is 99.95%. The threshold value that maximizes the F-Measure is $t_{CDnet} = 0.15$.

In addition to the statistics presented above, a qualitative evaluation is shown in Figure 3.6. Specifically, the segmentation output of our model is compared to the ground truth, while the background of the video sequence is also provided along with the input of the current frame. All 11 categories of the CDnet dataset are evaluated; from top row to bottom: Bad Weather, Camera Jitter, Shadow, Dynamic Background, Thermal, Baseline, Intermittent Object Motion, Turbulence, Low Framerate, Night Videos, PTZ. While black and white pixels indicate background and foreground respectively, the gray pixels of the segmentation images are outside of the Region Of Interest (ROI) and are ignored when calculating the evaluation metrics. Finally, the last row represents the Zoom-In-Zoom-Out sequence of the PTZ category, in which the background constantly changes because of the camera's zoom.

We also perform an analysis to further investigate the capability of the model to capture temporal features and its performance on completely unseen videos with different backgrounds. We formulate the experiment as follows: for every category of CDnet, we train on some videos and test on the rest. The complete train/test split and the full results are depicted on Table 3.3. As it can be seen, there is a positive FM score for every single category. This clearly demonstrates that the model is learning to segment moving objects to a good degree. More specifically, the model performs very well in categories where the backgrounds are similar. For example, all videos of the *Thermal* category have a grey, faded background and the proposed model achieves an average FM score of 0.8878. Similarly, the videos of the *Bad Weather* category are all covered in snow, and our model is reaching $FM = 0.5577$. The *Shadow* category is another prime example of our model's generalisation ability, since it reaches an average FM of 0.7207. On the other hand, the model's accuracy is very low on *Dynamic Background* and *Intermittent Object Motion*, as it has no context of which moving objects to segment. Finally, its performance is also suffering on *Night videos* and *Turbulence*, where the foreground objects are very small and there is a significant amount of noise.

Finally, failure cases are presented in Figures 3.7, 3.8. For each category, we calculate the image with the lowest F-Measure score. From the results it can be seen that the most common failure modes are either extremely small foregrounds and/or objects appearing at the edge of the image. In the *Intermittent Object Motion* the failure mode is caused by occlusion, while in *Night Videos* it is due to the car's headlight's glare, which is similar to that of the light poles in the background. Finally, in *PTZ* there are some false positives caused by the camera movement. We conclude that using multi-scale inputs or dilated convolutions with multi-scale kernels can alleviate the false negatives of small objects.

ESI

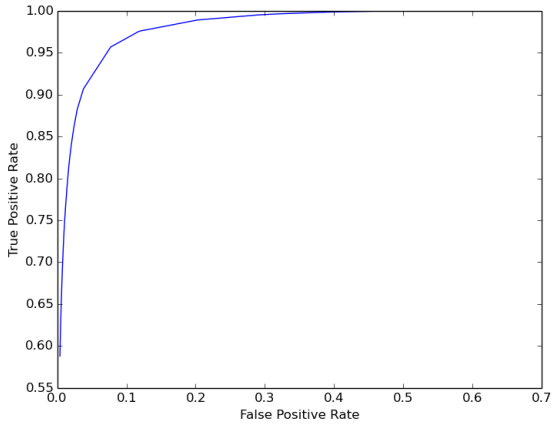
For the evaluation of our model in this dataset, we compare against the following models: Eigen background based Statistical Illumination (ESI) [135], Statistical Illumination (SI) [98], dynamic Eigen Background (EBdyn) [91], fixed Eigen Background (EBfix) [46], Tonal Alignment (TA) [125] and Adaptive Background Mixture Model (ABMM) [61]. It should be noted that most of these methods were designed specifically for background subtraction in environments featuring sudden illumination changes. As depicted in figure 3.4, we achieve higher performance than every other model. Full metric results in each video sequence are given in table 3.5. Our model performs best in Scene2, in which there is minimal movement of the foreground and therefore the model is well trained to recognize the foreground. Conversely, the lowest scores are reported in the video sequence of Chair, where the foreground varies significantly in each frame. Due to the highly limited number of frames, it was impossible for our model to be trained appropriately. In addition, since in some video sequences the frames in the train set vary significantly compared to those of the test set, a more sophisticated separation could lead to better results. Figure 3.3 depicts the ROC curve of our evaluation, which was generated using 30 threshold values within

$(1 \times 10^{-5}, 0.8)$. The optimal threshold value was $t_{ESI} = 0.3$, while the total area under the curve is 98.43%.

Table 3.5 F-Measure results on the ESI dataset

Scene	Recall	Precision	F-Measure
Walking	0.8073 ± 0.0124	0.7803 ± 0.0145	0.7915 ± 0.005
Sofa	0.761 ± 0.0097	0.8115 ± 0.0078	0.7725 ± 0.0047
Chair	0.7086 ± 0.0115	0.7648 ± 0.0101	0.7289 ± 0.0035
Scene1	0.8587 ± 0.0163	0.8171 ± 0.0145	0.8326 ± 0.0041
Scene2	0.9076 ± 0.0101	0.8823 ± 0.0149	0.8921 ± 0.0039
Overall	0.8021 ± 0.0137	0.7981 ± 0.0125	0.7994 ± 0.0044

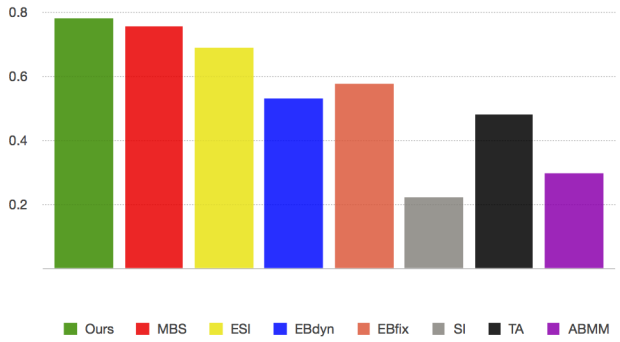
Fig. 3.3 ROC curve on ESI dataset



A qualitative comparison between our model and the state-of-the-art is presented in figure 3.5. In the first column input frames of different videos demonstrate the various stages of lighting featured in the dataset. The rest of the table shows that our model produces output extremely similar to the ground truth in all sequences and lighting conditions.

[2]The F-Measure values of the other models are taken from Sajid and Cheung [106].
[3]The segmentation pictures of ESI and MBS are taken from Sajid and Cheung [106].

Fig. 3.4 F-Measure values comparison on the ESI dataset [2]



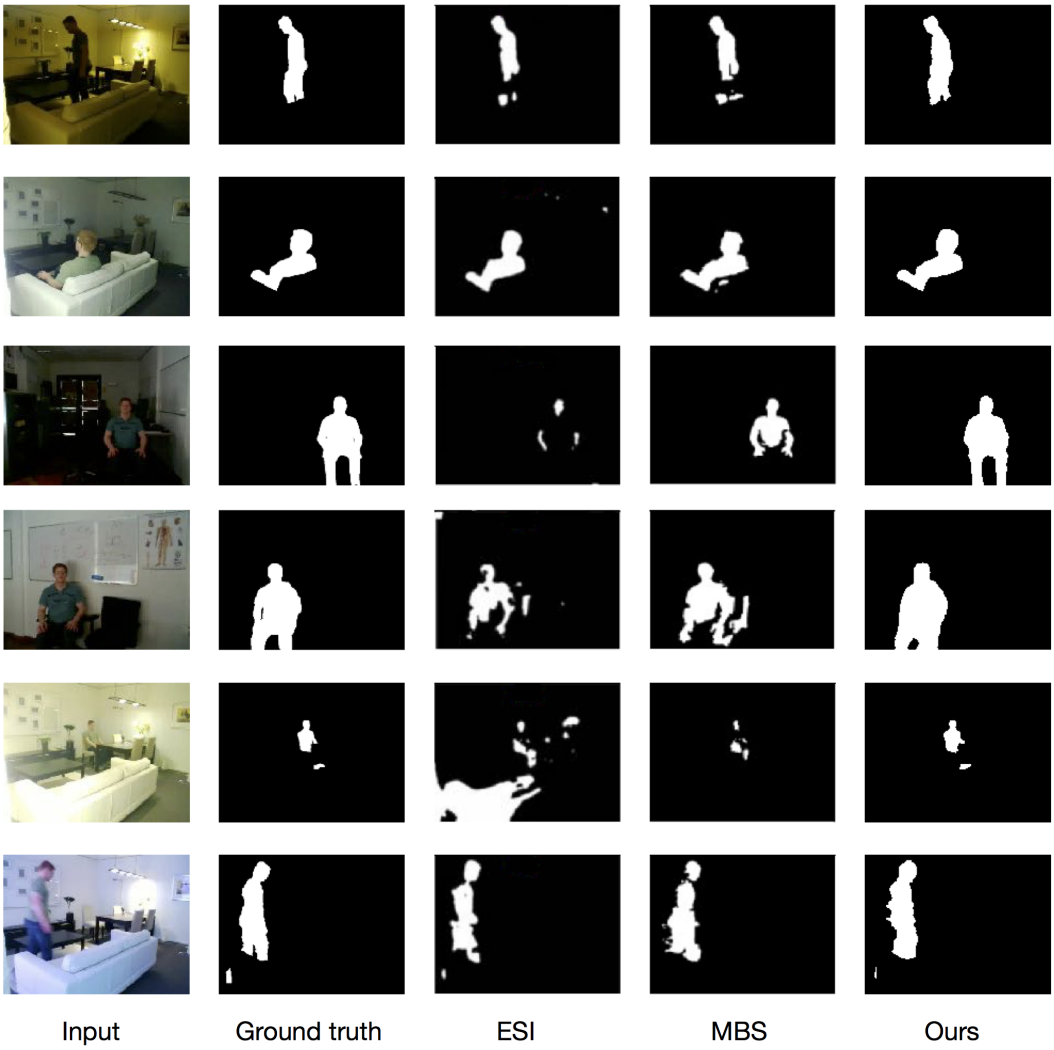
3.4 Conclusion

Most background subtraction approaches either use an additional system for background change monitoring, or ignore the aspect of time completely. In this work, we address these limitations and present a completely end-to-end temporal-aware approach for foreground segmentation with 3D convolutional neural networks. By simultaneously tracking changes on the spatial and temporal dimension, our model is able to effectively track the movement of the foreground and the relations between neighboring pixels using multi-modal features rather than depending on post-processing techniques. This is effectively accomplished by performing 3D convolutions on the 10 most recent frames of the video. Four upsampling layers of different kernel sizes utilize information from shallow, mid-level and deep layers of the network in a multi-scale approach, which leads to increased segmentation accuracy.

Experiments on two different and especially challenging datasets show that our model outperforms all other methods overall since we improve the state-of-the-art in every metric. We also show that our model can handle numerous different scenes very effectively.

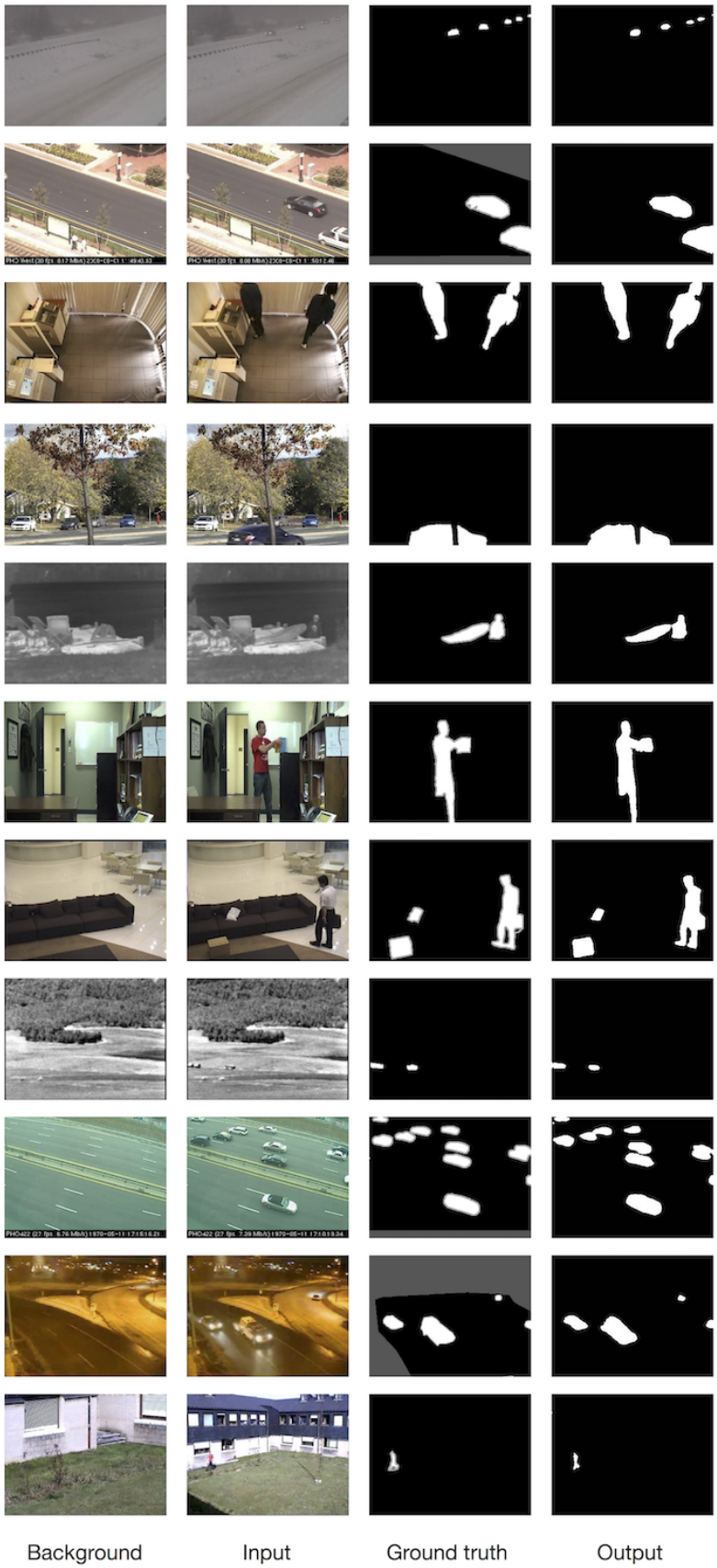
For future work, a detailed study can be performed regarding the optimal number of input frames as well as the stride of the temporal window. In this chapter, we focused on creating a single model that works in all scenarios. Most of the videos included in this dataset are of similar frame rate, with the exception of the *Low Framerate* category of the CDnet dataset.

Fig. 3.5 Segmentation results of ESI [135], MBS [106] and ours on ESI dataset. [3]



The frame rate of the videos belonging to that category varies from 0.17 frames per second (fps) to 1 fps due to limited transmission bandwidth [140]. In the future, an adaptive temporal window could be considered. For example, the stride could be increased in cases of videos with slow-moving objects. Therefore, the model input would be normalised in the temporal dimension. Additionally, this would lead to a decrease of input frames and therefore a reduction of computational costs. Furthermore, using a pre-trained model could boost results significantly, while a weighted loss function could offer additional improvements. Lastly,

Fig. 3.6 Segmentation results of our model in all categories of the CDnet dataset.



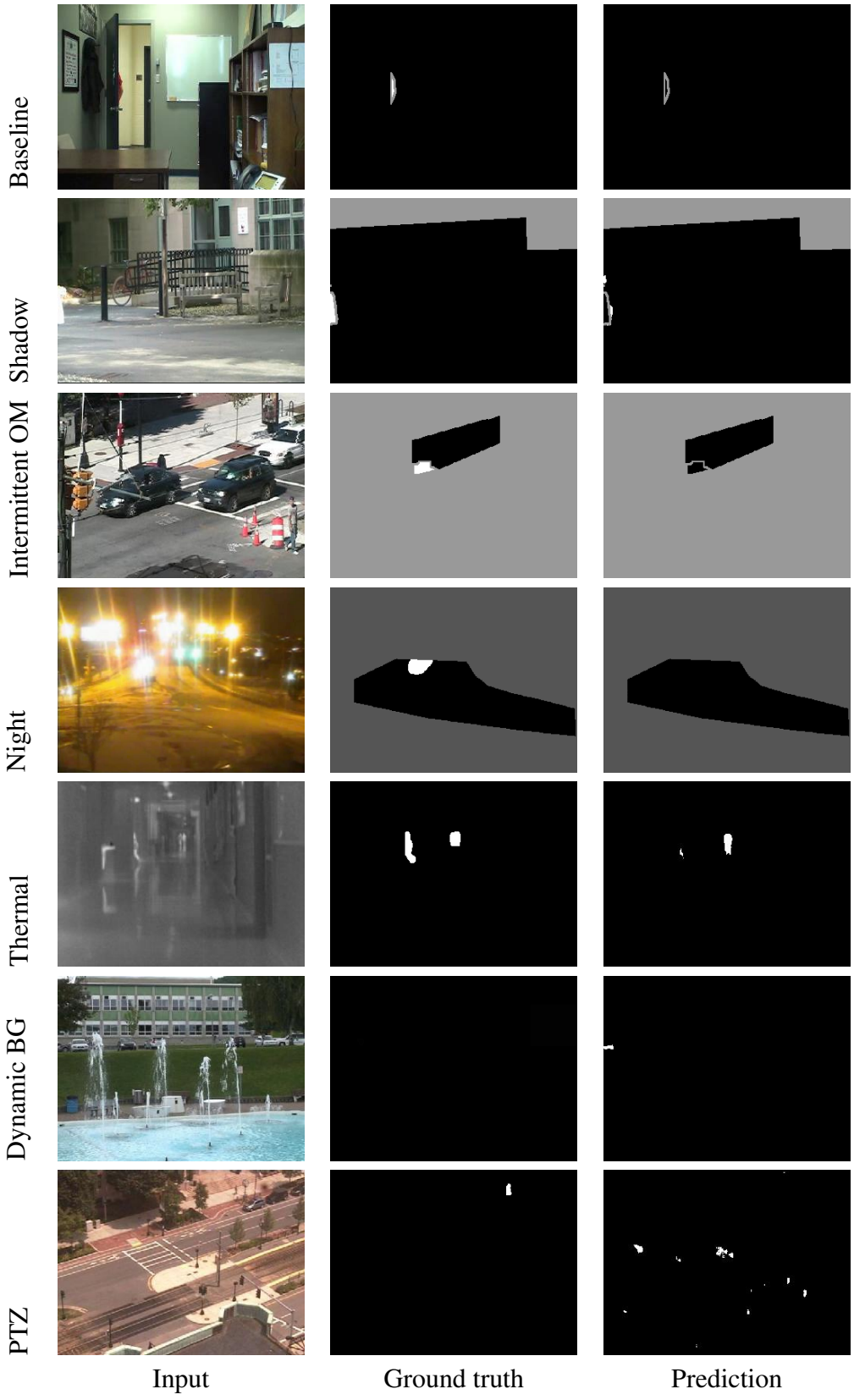


Fig. 3.7 Failure cases. Pixels in grey colour are outside of the region of interest.

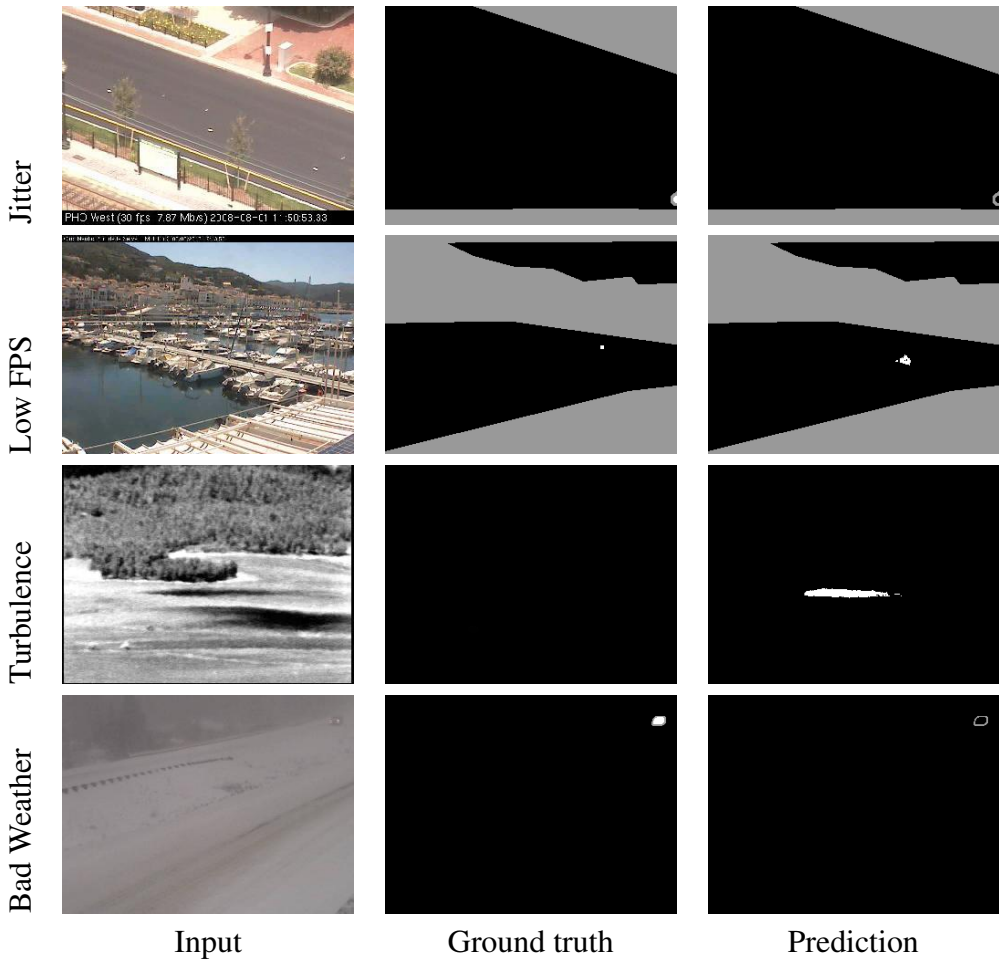


Fig. 3.8 Failure cases. Pixels in grey colour are outside of the region of interest.

implementing a better augmentation technique could be useful in extending the dataset and therefore reducing overfitting.

Chapter 4

Synthetic data augmentation for robust foreground segmentation

In the previous chapter, we addressed the vast majority of challenges present in the task of background subtraction. Our model achieved a very high score FM across a plethora of different scenarios. However, the model scored a lower FM on the dataset of illumination changes. As a matter of fact, sudden illumination changes signify a particularly difficult challenge, since they can cause drastic intensity alteration to the entire scene and are often unexpected. Such changes in lighting conditions can be caused either by weather conditions or malfunctioning of electric lights and result in color changes of a significant amount of pixels. Due to the difference of visual appearance in consecutive frames, BGS becomes inaccurate. The timing of these changes could be short, such as switching a light on/off, or a piece of cloud blocking the sun, making it tough for the system to adjust to the new condition in a timely manner.

In this Chapter, we address this challenge with a semantic data augmentation method, targeted for illumination changes. We evaluate our technique in the most challenging dataset for illumination changes in foreground segmentation, the *Stuttgart Artificial Background*

Subtraction (SABS). Experimental results show the proposed scheme is superior compared to traditional methods.

4.1 Introduction

State-of-the-art deep learning algorithms allow adapting to sudden illumination changes if a huge amount of training data is provided. Otherwise, such changes can affect the model performance drastically: flashes of light will increase the number of false positives, while shadows can increase the false negatives. However, obtaining labelled data is very costly and there is only limited datasets available in the community. As a solution, data augmentation methods are proposed to perform image-based operations on the data, such as mirroring or cropping, to synthesize a larger dataset. However, simple image tricks cannot effectively generate images with realistic illumination changes. Another solution is adding a small amount of noise to create a new, synthetic image that is similar to the original in context but different in color distribution. However, since the added noise does not have any semantic meaning, the synthetic images only slightly increase the generalisation power of the model, as they do not offer any additional knowledge of different lighting conditions of the same scene.

To overcome this challenge, we propose a new data augmentation technique by synthesising the light-based effects of different degrees of brightness. Such effects include shadows and halos of different size, placed in random locations of the input image. In addition, global illumination changes are also included, in order to increase the generalisation abilities of the model to scenes filmed at various times of the day and night. Such augmented data allows us to provide extra semantic information to the BGS model in terms of illumination for better generalisation performance. The results show that the proposed technique is superior to

regular augmentation methods and can significantly boost the segmentation results even in scenes that feature illumination conditions unseen to the model.

Furthermore, a post-processing method is proposed that can successfully remove noise from the output binary map. The method is based on the fact that contiguous frames have minimal changes between them and thus, the potential areas of the output that include foreground objects can be limited. Our experiments indicate that the proposed method can provide further improvements.

The main contributions of this work can be summarized as follows:

- A novel synthetic image generation method for robust foreground segmentation under challenging illumination conditions.
- An effective deep neural network for foreground segmentation.
- A post-processing technique based on temporal coherence for the refinement of the segmentation results.

The remainder of this chapter is organised as follows. Section 4.2 outlines the proposed method for synthetic generation covering local, global and combined changes. The post-processing technique is also explained. Section 4.3 follows with the presentation of our results and discussion. Finally, Section 4.5 provides the conclusion and future work.

4.2 Methodology

This section presents our methodology. As mentioned above, we synthesise both local and global changes, and then we combine them to a unified augmentation method that covers all scenarios simultaneously. In the following subsections we explain in detail the creation of the synthetic images for all cases. In addition, we describe the post-processing method for the output refinement.

4.2.1 Local changes

For the case of local changes, we generate the synthetics by locally altering the illumination of the input image, therefore creating either a "lamp-post" or a shadow effect. First, we randomly select a pixel of the image which serves as the centre of the circle to be drawn: $p = I(w,h), w \in W, h \in H, I = W \times H$, where W, H the width and height of the input image I respectively. Once the coordinates of the centre pixel are determined, we need select the diameter d of the circle. Naturally, we want our model to be robust to both small and large shadows and flashes of light, therefore various values of d can be randomly selected during training. However, scaling is very important to produce a realistic effect. In scenes filmed from afar, d is expected to be smaller - conversely if all objects appear larger, the value of d should be adjusted accordingly. Therefore $d \in [d_{min}, d_{max}]$ is left as a hyper-parameter.

Furthermore, if we were to modify all pixels within the circle evenly, the result would be far from realistic. Therefore, a more sophisticated approach is adopted. First, we calculate the binary mask M_1 of the pixels to be altered using the following formula:

$$M_1(x,y) = 1 \Leftrightarrow (x-w)^2 + (y-h)^2 \leq d^2 \tag{4.1}$$

Therefore, the pixels of our mask will have the value of 1 if they reside within the drawn circle and zero everywhere else. For the next step, we use the Euclidean Distance Transform (EDT) which, given a binary mask B, can be defined as follows:

$$EDT_x(B) = min_b(||x-b||_{L_2}), \quad \forall b \in B, \tag{4.2}$$

where L_2 is the Euclidean norm. Now, we can calculate the mask for local changes M_2 by applying the EDT on M_1:

$$M_2 = EDT(M_1) \tag{4.3}$$

Once the new mask has been created, we proceed to alter the pixels of the original image that lie within the circle. Therefore, the new synthetic image I_s is calculated as follows:

$$I_s = I \pm (M_2 \times z_l), \quad z_l \in [l_{min}, l_{max}], \tag{4.4}$$

where I the original image, M_2 the mask calculated with the distance transform, z a random integer, and $\pm$ is either pixel-wise addition or subtraction, chosen with probability $p = 0.5$.

The application of the aforementioned local masks are depicted in Figure 4.1. Clearly, the final lamp-post effect looks realistic.

4.2.2 Global changes

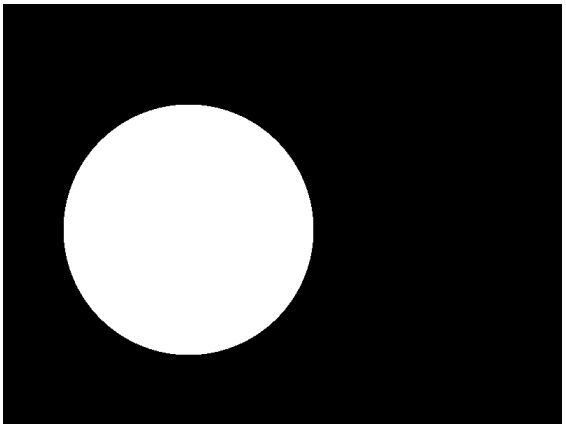
In some cases, global illumination changes can occur. For example, a light might be switched off suddenly. In outdoor scenes, a lightning during a storm may instantly increase the brightness, and once the rain is over the global illumination will change again. In order to model such illumination changes, we need to alter the pixels of the whole image, rather than a small patch.

Therefore, the new synthetic image is generated as follows:

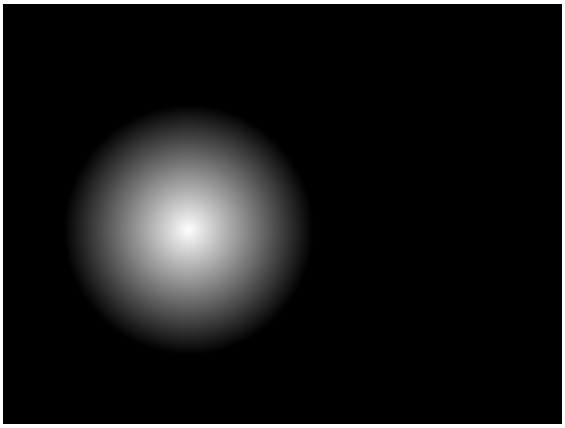
$$I_s = I \pm z_g, \quad z_g \in [g_{min}, g_{max}], \tag{4.5}$$

where I, z and $\pm$ are as previously defined.

In this case the illumination noise z needs to be slightly diminished, since the whole image is affected.



(a) The mask M_1



(b) The mask M_2

(c) Original image

(d) After effect

Fig. 4.1 The application of the mask for local changes. Subfigure (a): the initial binary mask M_1 is created by a circle of diameter $d = 179$ and centre coordinates $(322, 265)$. Subfigure (b): The mask M_2 after the application of the Euclidean distance transform on M_1. Subfigures (c) and (d) depict the original image and the lamp-post effect after the application of the mask M_2 on the input image respectively.

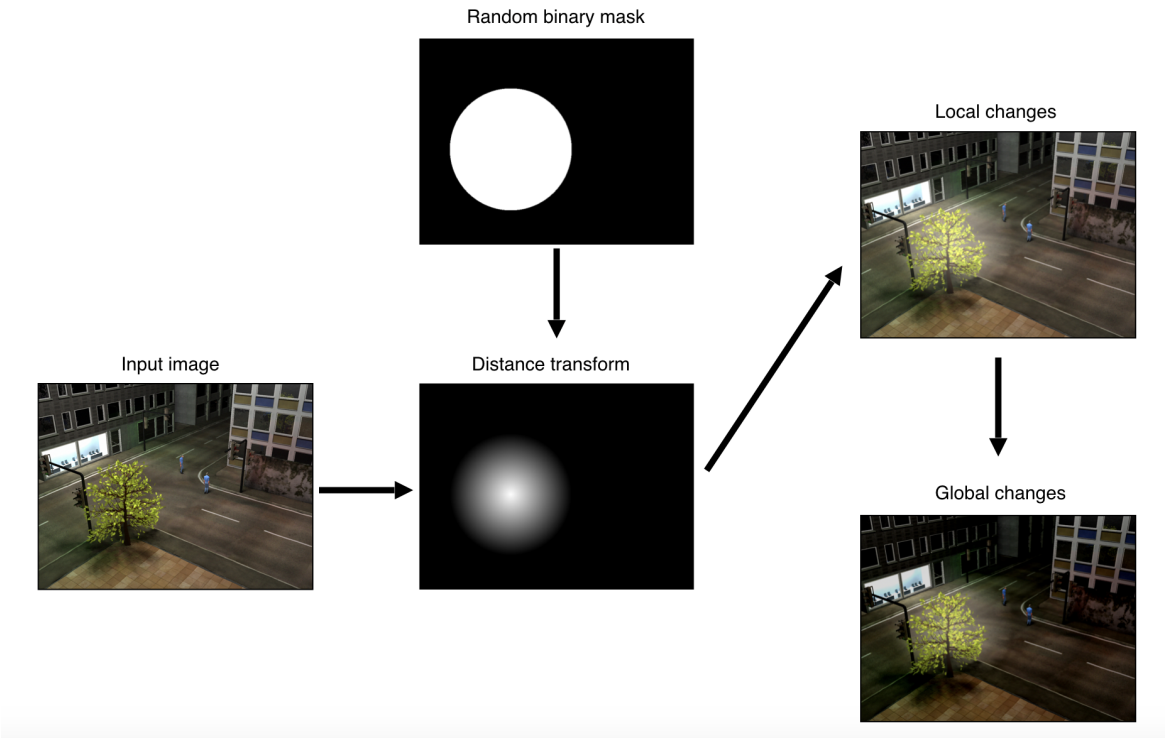


Fig. 4.2 Methodology

4.2.3 Combined changes

To capture both local and global illumination changes in the scene, we combine equation 4.4 and equation 4.5 into the following:

$$I_s = z_l \pm (I \pm (M_2 \times z_g), z_l \in [l_{min}, l_{max}], z_g \in [g_{min}, g_{max}] \tag{4.6}$$

Sample images synthesised from our system can be found in figure 4.3. Since both the positioning and the intensity of the masks is random, this method can effectively cover all kinds of illumination changes. Additionally, hundreds of different synthetic images can be generated from a single frame. Therefore, given a small video, we can generate enough unique synthetic images to train a very deep network. The full methodology is depicted in Figure 4.2.

(a) G_bL_b

(b) G_bL_d

(c) G_dL_b

(d) G_dL_d

Fig. 4.3 Combination of global and local illumination changes. The subfigures (a) and (b) depict a combination of a brightening global filter with a bright and dark local filter respectively. On the other hand, subfigures (c) and (d) implement the darkening global filter.

4.2.4 Output refinement via temporal coherence

While the proposed augmentation method works for still images and videos alike, in the latter case we can exploit the motion information to refine the segmentation results. As long as the frame rate is not extremely low, the following hypothesis holds true:

Lemma 1 *Let $o_t = (i, j)$ be a pixel of an object at time t. Then the corresponding $o_{t+1} \in \{(i \pm \delta i, j \pm \delta j)\}$, where δ is a small integer.*

Based on the above, we can create a probability map p_t to highlight the areas of the input image that are likely to contain pixels of the foreground in the next frame. The map will act as a weight matrix that will refine the probabilities of each pixel of the model output. Essentially, p_t first needs to ensure that the foreground probability of those pixels of s_{t+1} that belong to the foreground will not be scaled down. This is a desired property, since the change between two contiguous frames is minimal and most foreground pixels will remain in the same class. Secondly, those pixels of p_t which are adjacent to foreground pixels need to be assigned with a probability value very close to 1, as it is highly possible for the foreground object to move into this area. As the distance becomes larger, the values will need to be gradually scaled down. Eventually, the pixels that are furthest away from the foreground will have the smallest probability.

We can construct p_t in the following way: given each timestamp t and a video frame F_t, we obtain the model output s_t, the pixels of which represent the probability of them belonging to the foreground class. Then, p_t can be generated by applying the euclidean distance transform on s_t:

$$p_t = EDT(s_t), \tag{4.7}$$

where EDT is as defined in equation 4.2.

While this is a valid approach for existing moving objects, we need to account for new objects entering the scene at any moment. As a result, we set the values of p_t located around

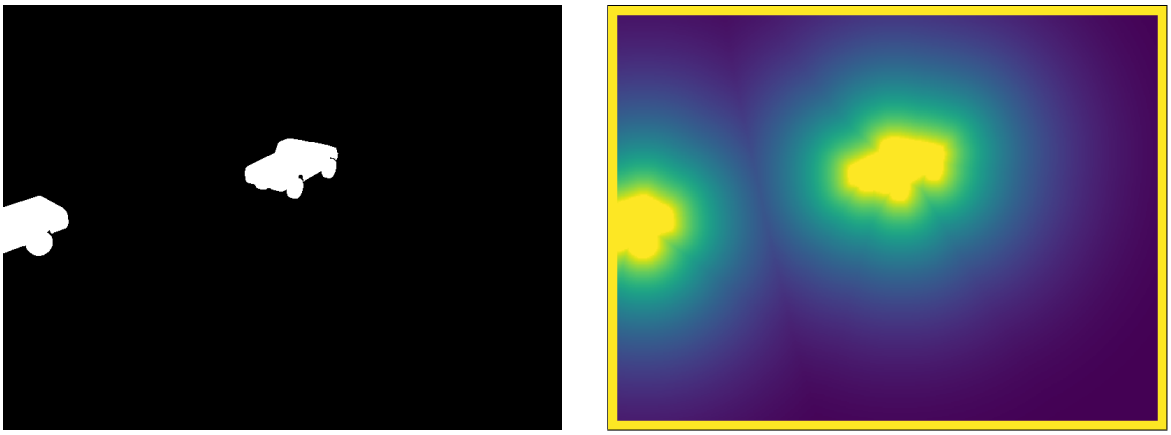


Fig. 4.4 The probability mask that is used for refining the output, created from the segmentation mask of the previous frame. Bright colours indicate high probability, whereas dark colours show low probability values.

the border to 1. Therefore, the mask will not penalise new objects entering the frame. The end result of the probability mask p_t is depicted in figure 4.4.

Having defined the process of creating the probability map, the refinement is performed in a post-processing manner. During testing, we obtain the model output of the current frame and calculate the probability map, which is used to filter the model output on the next frame. Thus, s_{t+1} can be refined by scaling its probability values according to p_t as follows:

$$s^r_{t+1} = s_{t+1} \times p_t, \tag{4.8}$$

where $\times$ is the pixel-wise multiplication operator and s^r_{t+1} denotes the refined segmentation result.

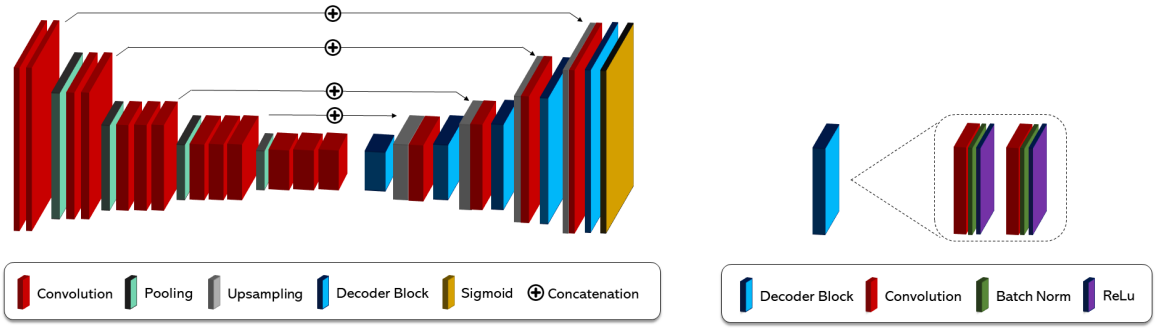
4.2.5 Illumination-invariant Deep Networks

We utilise the synthetic images to train multiple deep learning networks for BGS and evaluate their performances. Due to the use of images with synthetic illumination changes, these networks become invariant to lighting conditions.

To ensure the fairness of our experiments, all models used in this study have the same architecture. We follow the paradigm of previous foreground segmentation approaches [71, 157] and use a Unet architecture, which comprises an encoder and a decoder. We employ transfer learning and use the VGG16 model [115] as the encoder. Therefore, the weights of the encoder are initialised from those of VGG16, which has been pre-trained on Imagenet. VGG16 encompasses 13 convolutional layers, 5 pooling layers and 3 fully connected layers. Following the work of Long et al. [82], we remove the fully connected layers and make the network fully convolutional. Because of the pooling layers, the output of the encoder is 5 times smaller than the input. We use the decoder to recover the information that is lost from the downsampling operation via the use of upsampling blocks. Each block consists of a 2x2 bilinear interpolation operation which upsamples the feature maps, followed by two 3x3 convolutional layers with batch normalisation applied in-between (Figure 4.5b). To maximise the information recovered by the encoder, we add skip connections that connect the encoder to the decoder. In addition, the ReLu non-linearity is applied after each convolutional layer. Finally, once the spatial size has been restored, we add a final 3x3 convolutional layer, followed by a sigmoid layer to convert the output of the model to a foreground probability map. The architecture of the network is given in Figure 4.5.

4.2.6 Training settings

Here we describe the training process of the CNNs. All models are trained with the same parameters. The initial learning rate is $lr = 0.001$ and is reduced by a factor of 0.1 if the model does not improve for 2 epochs. The training process ends after 5 epochs of no improvements. For optimisation, the Adam optimiser [65] is selected with betas $b_1 = 0.9$ and $b_2 = 0.999$. Finally, the batch size is set to 1.



(a) Overview (b) Decoder block

Fig. 4.5 The CNN that was used for the experiments. The encoder is initialised from VGG16 [115] and is keep fixed during training. ReLu layers are used after every convolution and are omitted from Figure (a) for clarity. Features of the encoder are concatenated with those of the decoder which are of the same size, to enable information flow.

To avoid overfitting, we freeze the encoder of our network. Specifically, the first 5 convolutional blocks of VGG16 are fixed and we only train the decoder. This training procedure yields better results according to our experiments.

Furthermore, for all augmentors, the probability of each training sample being augmented is set to 66.7%.

Most frames contain many more pixels of the background than the foreground - some frames might not depict any moving objects at all. Given this observation, the loss function needs to balance the classes as to not allow the model to be biased towards the background class. Therefore, we use the weighted cross-entropy loss, which is formally defined as follows:

$$G_s = w_f t[-\log \sigma(x)] + w_b(1-t)[-\log(1-\sigma(x))], \tag{4.9}$$

where w_f, w_b are the class weight coefficients, x is the predicted label, t is the target label and $\sigma(x) = \frac{1}{1+e^{-x}}$ is a sigmoid function. The class weight are calculated according to the

ground truth frames with the following formula:

$$w_i = \frac{N}{2 \times N_i}, i \in \{b, f\} \tag{4.10}$$

where N denotes the number of pixels of all input frames and N_b,N_f are those pixels that belong to the background and foreground respectively.

4.2.7 Implementation details

We use the Keras library [27] for training our models. In addition, for the quick deployment of the proposed model, the *Segmentation models* [148] library is used. The Graphics Processing Unit (GPU) that was used in all our experiments is an NVIDIA Tesla K80.

4.3 Evaluation

This Section describes the metrics that were used in this study and presents the results of our experiments.

4.3.1 Metrics

For evaluating our experiments, we use the same metrics that were used in Chapter 3, as defined in equations 3.4-3.10. In addition, to directly compare with other models which use other metrics, we also use *Matthews correlation (MC)* and *Intersection over Union (IoU)*, which are formally defined as:

$$MC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP) \times (TP+FN) \times (TN+FP) \times (TN+FN)}} \tag{4.11}$$

$$IoU = \frac{TP}{TP+FP+FN} \tag{4.12}$$

where TP, TN, FP, FN denote the true positive, true negative, false positive and false negative pixels respectively.

4.3.2 Dataset

In order to highlight the robustness of our augmentation process, we select the Stuttgart Artificial Background Subtraction dataset (SABS) [15].

The SABS dataset [15] contains 9 synthetic video sequences. The main challenge of the dataset stems from the sudden change of illumination over time. Although the foreground movements are the same in some sequences, the illumination is changing over time. In addition, different videos have very different lighting conditions, such as day-time and night scenes. In our experiments the sequence *Darkening* is used for training our models, which consists of 800 frames. The illumination of this scene is gradually changing from evening to night. For testing, the *Light Switch* video is used, which comprises 600 frames. This sequence only has night scenes and features light-switch effects in the middle of the video, where a store light is suddenly switched off. Since this effect is not present in the training video, *Light Switch* is an excellent candidate for measuring the generalisation abilities of our trained models. The rest of the video sequences in the SABS dataset [15] are not used because they are either day-time scenes and/or do not have significant illumination changes over time. Some example frames of the training and testing set are depicted in figure 4.8.

4.4 Results

We perform extensive evaluations on the proposed method. In particular, a wide range of different augmentation settings (Table 4.1) were evaluated. We also compare against the regular augmentation techniques. We implement a "default" augmenter which performs the following image transformations: *horizontal flipping, random cropping* and *noise addition*,

Fig. 4.6 Regular augmentation techniques (from left to right): image mirroring, center cropping and adding noise.

as depicted in Figure 4.6. The *cropping* operation performs center cropping with random image sizes, whereas the *noise* option adds salt and pepper noise drawn from a Gaussian distribution. The amount of noise is fixed to 0.05. All operations have a 50% probability of taking place.

In the following section, we will evaluate the proposed method quantitatively to determine the optimal settings.

4.4.1 Quantitative Evaluations

In this experiment, we evaluate the performance of the proposed method on the SABS dataset using the commonly used metrics stated in section 4.3.1. The results of our experiments are presented in Table 4.2. Even though the default augmenter improved the F-Measure by about 6%, the proposed model, named *GL*, outperformed it by a very large margin which amounts to almost 14%. As a matter of fact, the proposed method obtains better results in every single metric. This highlights the effectiveness of our proposed method.

In addition, to determine the optimal settings, an ablation study is conducted and the details are explained in Section 4.4.3.

Name	**Description**	**Threshold**
baseline	No augmentation	0.8
default	Common augmentation: Mirror, crop and noise	0.7
L_a	Local changes with $(l_{min}, l_{max}) = (80, 120), (d_{min}, d_{max}) = [\frac{X}{2}, \frac{2X}{3}]$	0.7
L_b	Local changes with $(l_{min}, l_{max}) = (80, 120), (d_{min}, d_{max}) = [\frac{X}{5}, \frac{X}{2}]$	0.7
L_c	Local changes with $(l_{min}, l_{max}) = (120, 160), (d_{min}, d_{max}) = [\frac{X}{5}, \frac{X}{2}]$	0.6
G_{low}	Global, low intensity changes with $(g_{min}, g_{max}) = (20, 60)$	0.9
G_{med}	Global, medium intensity changes with $(g_{min}, g_{max}) = (40, 80)$	0.6
G_{high}	Global, high intensity changes with $(g_{min}, g_{max}) = (60, 100)$	0.8
GL	Global and local changes with $(g_{min}, g_{max}) = (40, 80), (l_{min}, l_{max}) = (120, 160), (d_{min}, d_{max}) = [\frac{X}{5}, \frac{X}{2}]$	0.7
GL_{refine}	The GL model, after applying the post-processing method	0.6

Table 4.1 The different augmentation settings that were tested in our experiments. Parameters k, z and X denote the kernel size of the mask M_1, the illumination intensity in terms of pixel values and the resolution of the smallest dimension of the input image respectively. The last column shows the threshold that binarises the model output and which maximised the F-Measure of the segmentation mask. The GL_{refine} model employs the post-processing method described in Section 4.2.4.

Metric \ Setting	No augm	Common augm	GL	GL_{refine}
Recall ↑	0.4682 ±0.0255	0.5651 ±0.0212	0.7847 ±0.0189	**0.816** ±0.0142
Sp ↑	0.9932 ±0.00039	0.9936 ±0.00021	**0.9940** ±0.00027	0.9939 ±0.00018
FPR ↓	0.0069 ±0.00028	0.0063 ±0.00029	**0.0057** ±0.00019	0.0067 ±0.00017
FNR ↓	0.5583 ±0.00325	0.4744 ±0.00318	0.2292 ±0.00276	**0.2031** ±0.00288
PWC ↓	1.8947 ±0.0105	1.6484 ±0.0096	1.1284 ±0.0086	**1.1085** ±0.0088
FM ↑	0.5146 ±0.0083	0.6284 ±0.0081	0.7669 ±0.0072	**0.7752** ±0.0074
Precision ↑	0.6084 ±0.0211	0.6674 ±0.0196	**0.7468** ±0.0177	0.7326 ±0.0184
IoU ↑	0.3427 ±0.0195	0.4439 ±0.0189	0.6181 ±0.0155	**0.6278** ±0.0145
Matthews ↑	0.5180 ±0.0168	0.6049 ±0.0171	0.7591 ±0.0149	**0.7691** ±0.0152

Table 4.2 Comparison between no augmentation, common augmentation and the proposed method which covers global and local illumination changes. Best scores are highlighted in bold.

4.4.2 Qualitative Evaluations

To visualize the results, the segmentation comparisons between the models L_a, L_b and L_c are depicted in Figure 4.10. By comparing L_a to L_b, it can be seen that using smaller kernels removes a significant amount of noise that is present all over the image. The improvement is especially evident at the areas which underwent minor illumination changes, such as the windows of the buildings. This is an indication that local changes of low intensity can be handled by these kernels, but not those of large intensity, suggesting that the values of l_{min}, l_{max} need to be increased. Indeed, it is clear that the model L_c which is trained with higher l values can filter out the vast majority of noise caused by low intensity changes and a large portion of the highest level changes as well.

The effectiveness of the proposed approach is very clearly illustrated in Figure 4.11, where we further compare different augmentation settings. For example, the baseline model produces a very large amount of noise, as it is sensitive to the illumination changes caused by the traffic lights. In addition, it completely fails to segment the car when the light of the shop window switches off. Regular augmentation techniques definitely help to alleviate this issue, but only when the light is still on. Conversely, it can be seen that the model *GL* which was trained with the data generated by the proposed method successfully segments the car in all conditions, while also keeping false positives to a minimum.

4.4.3 Ablation studies

Here we discuss the ablation studies that verify the optimal hyper-parameters of our method. The full list of our experiments can be found in Table 4.1, whereas the result of each method is shown in Table 4.3. We also report the optimal threshold that needs to be applied to the model output, which is a probability map, to obtain the binary (foreground/background) mask. By cross-checking the aforementioned tables, it can be seen that using a smaller kernel

is better for creating local effects. Also, greater changes in illumination yield better results. This is because of the fading effect caused by the distance transform, which is only strong in the centre of the circle. For global changes a much smaller noise value is needed, firstly because there is no fading and secondly due to the effect being applied to the whole image.

It is evident from Table 4.3 that both local and global change augmentations yield significant results. However, the best performing model according to our experiments encompasses both.

It is noteworthy that sub-optimal settings need a very high threshold to produce a good segmentation result. This is because in those frames of the *Light Switch* sequence where the light suddenly switches off, the model fails to identify the moving object due to low lighting. However, with the optimal settings of the proposed method, the model can generalise in all illumination conditions. The performance of each model under different threshold is depicted in Figure 4.7.

4.4.4 Failure cases

We demonstrate some examples of failure cases in Figure 4.9. Three main problem-causing scenarios can be clearly identified. In the first example, we have an occlusion case, where the car is hiding behind the tree. Some car pixels are correctly classified, however there are many false negatives. The second example presents many false positives, which are caused when the light of the store is switched off. The incorrectly classified pixels belong to the tree, which not only appears much brighter compared to the other objects of the scene, but also is moving. Finally, the last example is the most challenging as it combines the first two. In addition, the foreground objects have darker colours, hence the contrast is much lower which leads to the model being unable to accurately segment the cars' boundaries. In this case we have both false negatives and false positives.

Metric	L_a	L_b	L_c
Recall ↑	0.5582 ±0.0245	0.5849 ±0.0261	0.6373 ±0.0189
Sp ↑	0.9957 ±0.00022	0.9946 ±0.00032	0.9955 ±0.00025
FPR ↓	0.0035 ±0.00015	0.0048 ±0.00019	0.0044 ±0.00024
FNR ↓	0.4516 ±0.00304	0.4108 ±0.00364	0.3686 ±0.00299
PWC ↓	1.4386 ±0.0128	1.4202 ±0.0113	1.3216 ±0.0149
FM ↑	0.6448 ±0.0089	0.6645 ±0.0098	0.6940 ±0.0067
Precision ↑	0.7730 ±0.0149	0.7498 ±0.0195	0.7744 ±0.0179
IoU ↑	0.4830 ±0.0229	0.5073 ±0.0207	0.5311 ±0.0168
Matthews ↑	0.6498 ±0.0137	0.6621 ±0.0191	0.6891 ±0.0155

(a) Ablation studies for local changes

Metric	G_{low}	G_{med}	G_{high}
Recall ↑	0.6853 ±0.0233	0.7167 ±0.0219	0.6795 ±0.0225
Sp ↑	0.9918 ±0.00037	0.9949 ±0.00045	0.9956 ±0.00019
FPR ↓	0.0076 ±0.00019	0.0064 ±0.00027	0.0048 ±0.00024
FNR ↓	0.2925 ±0.00373	0.2875 ±0.00264	0.3282 ±0.00295
PWC ↓	1.4127 ±0.0123	1.2487 ±0.0093	1.2414 ±0.0145
FM ↑	0.6948 ±0.0101	0.7228 ±0.0083	0.7167 ±0.0096
Precision ↑	0.6939 ±0.0117	0.7320 ±0.0131	0.7821 ±0.0161
IoU ↑	0.5413 ±0.0203	0.5687 ±0.0254	0.5578 ±0.0178
Matthews ↑	0.6924 ±0.0165	0.7196 ±0.0189	0.7131 ±0.0152

(b) Ablation studies for global changes

Table 4.3 Ablation studies for local and global changes

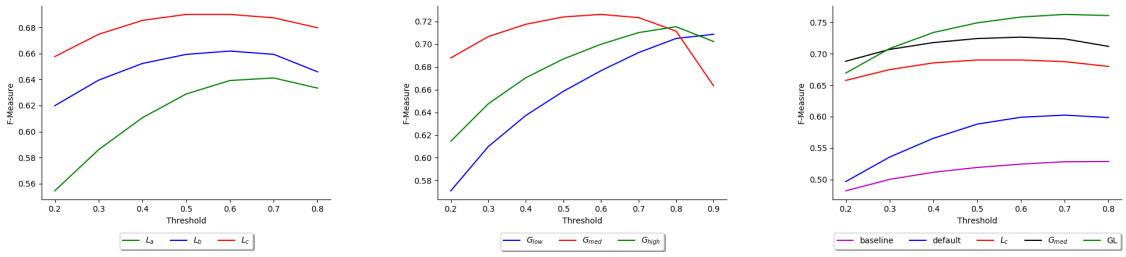


Fig. 4.7 F-Measure values on different thresholds for each model.

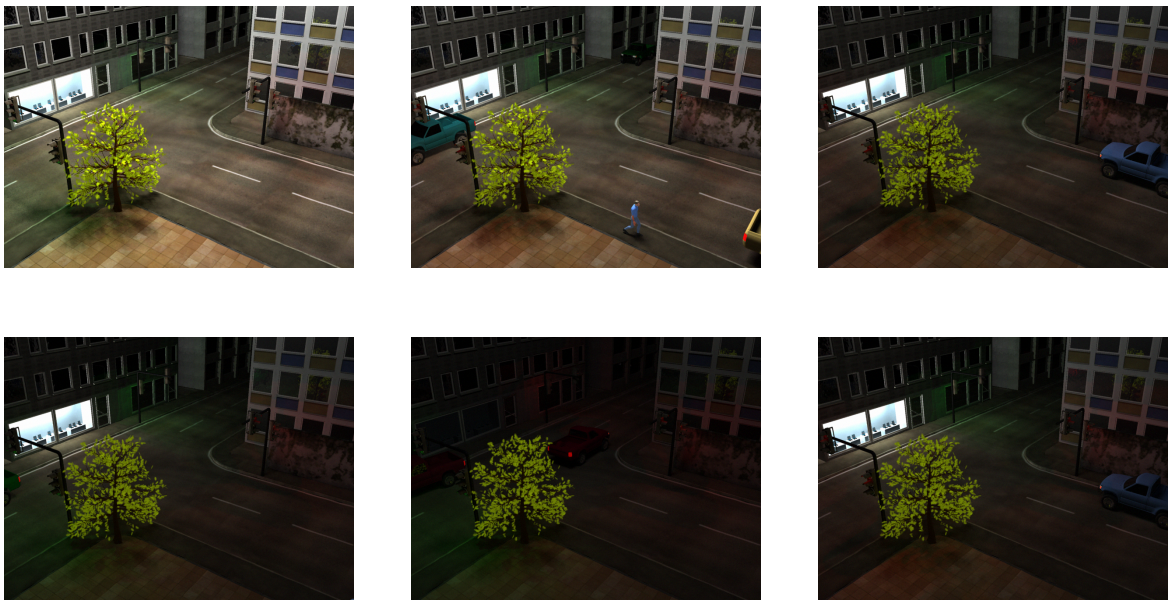


Fig. 4.8 The SABS dataset that was used for evaluating the models. The first row depicts the training sequence *Darkening*, while the second row shows the testing video *LightSwitch*. The columns show frames from the start, middle and ending parts of the video. Note that in the middle of the *LightSwitch* sequence the store light switches off, causing major changes to the background.

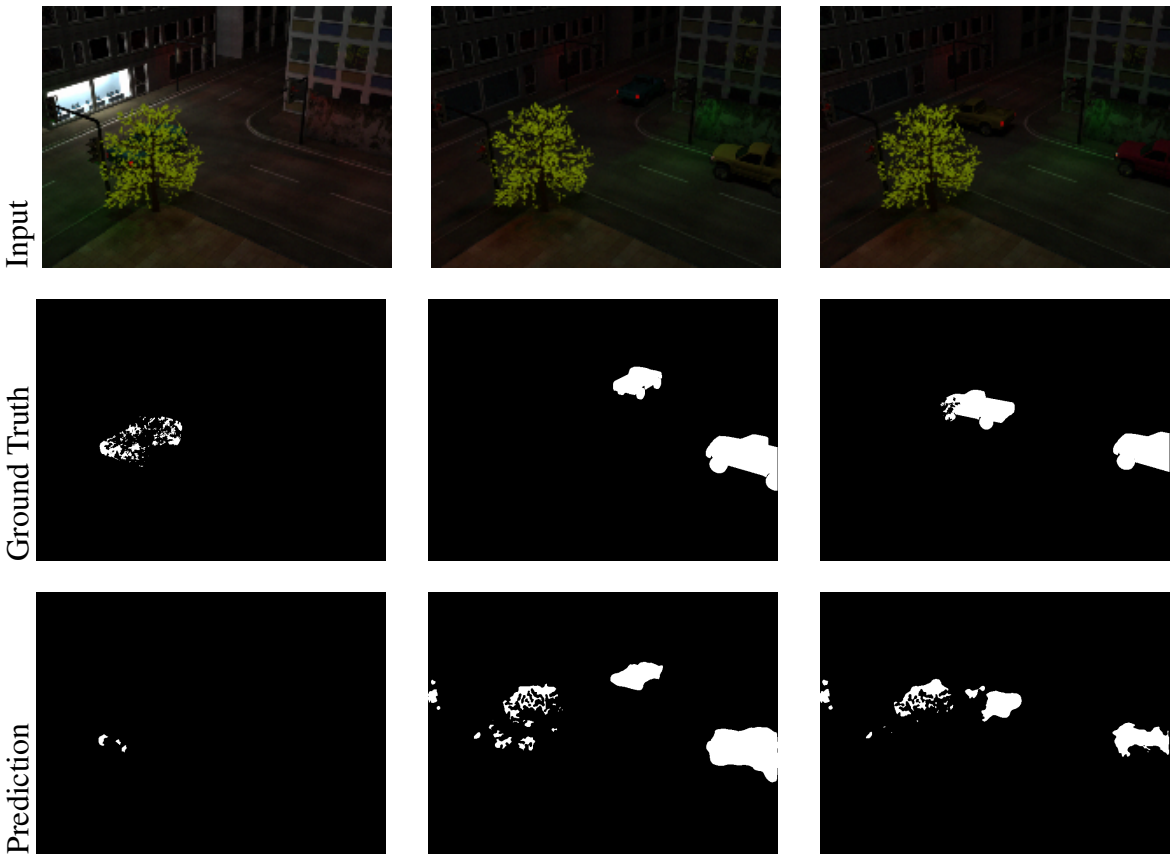


Fig. 4.9 Failure cases

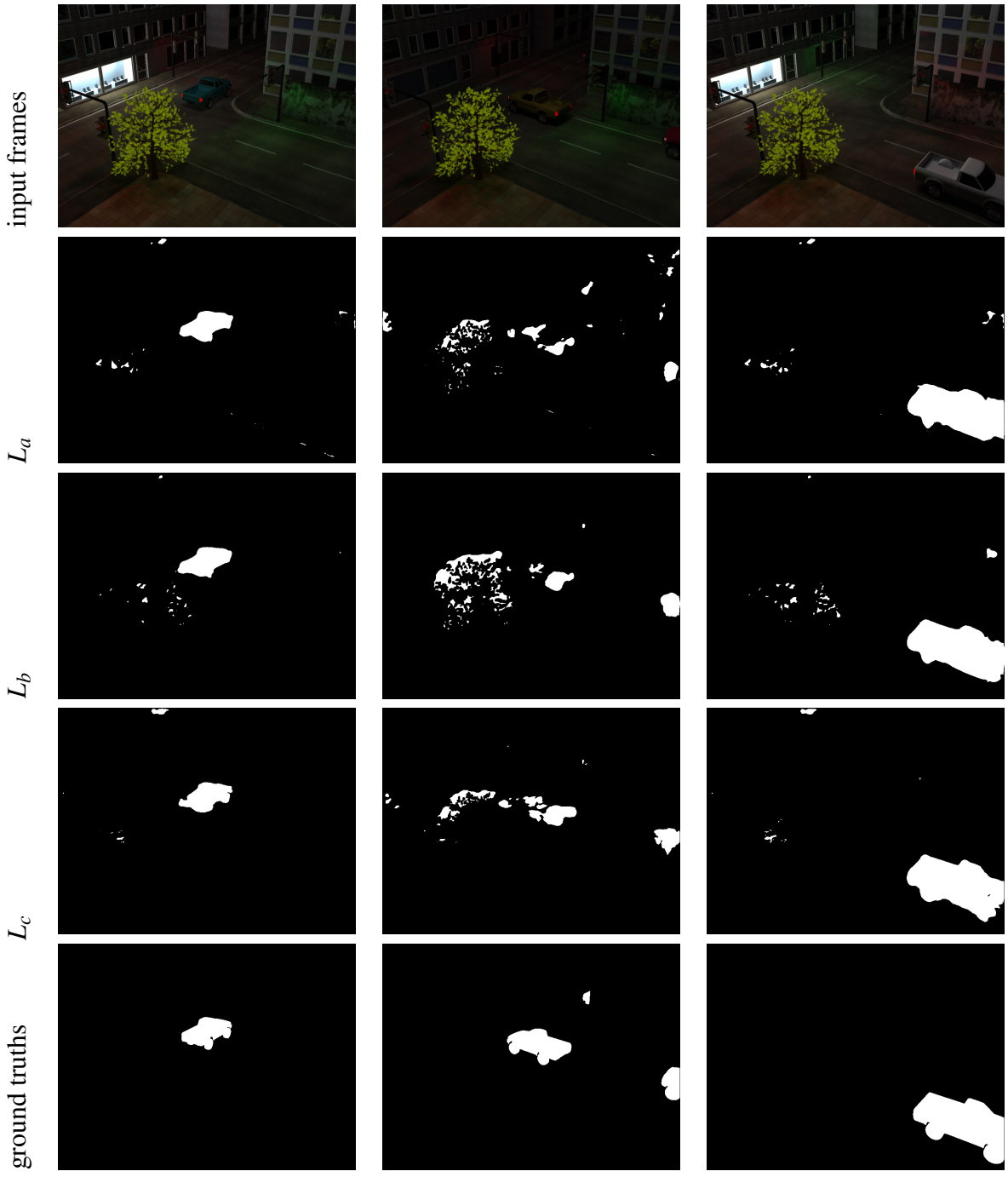


Fig. 4.10 Results with various kernels and intensities for the application of local changes.

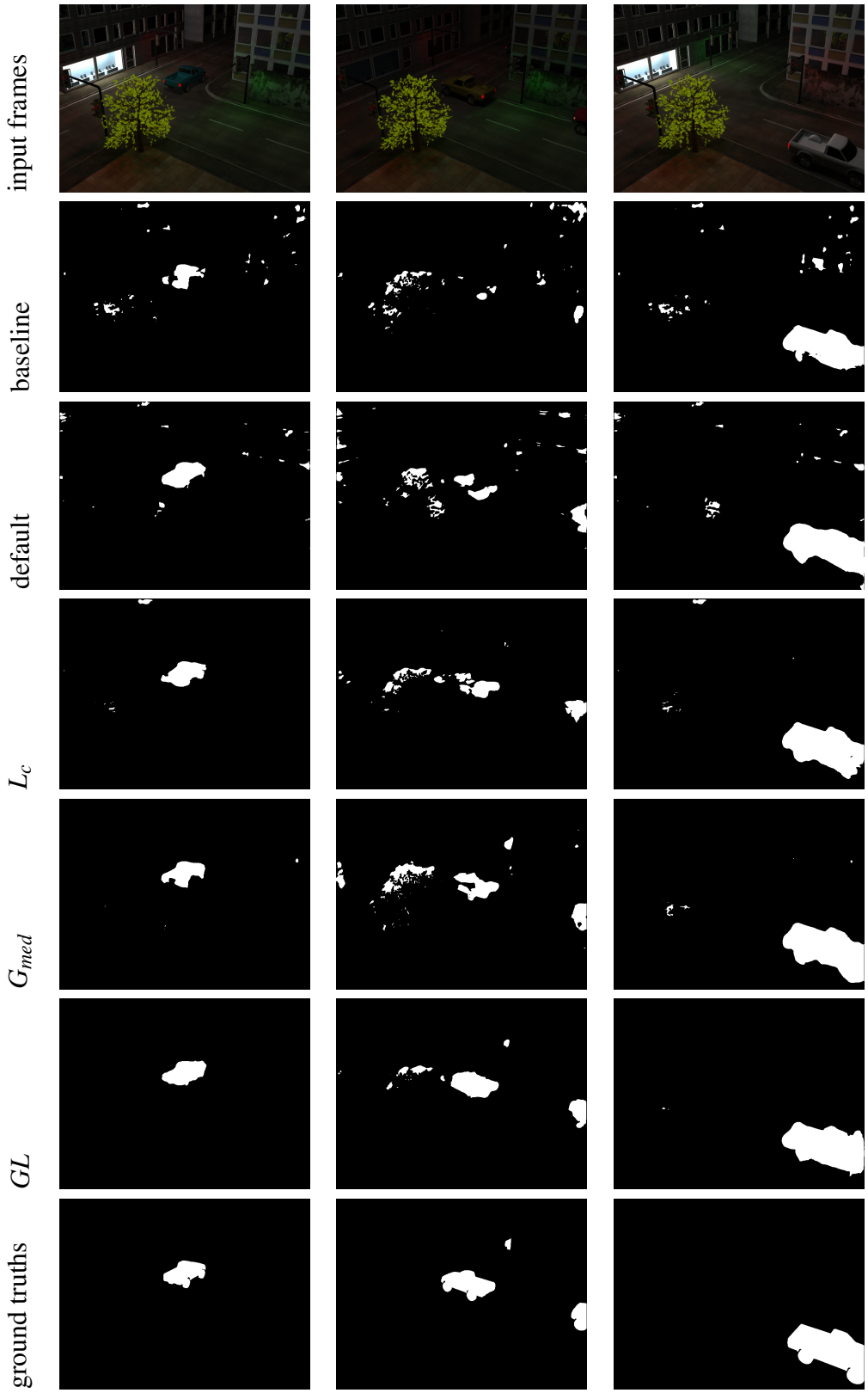


Fig. 4.11 Comparison between different augmentation techniques.

4.5 Conclusion and Future Work

In this Chapter, we have presented a fast and easy method to synthesise training samples for the implementation of illumination-invariant models. The synthetic images are generated by artificially altering the pixel intensity values of the input image not only globally but also in small regions. A typical "lamp-post" light source effect can be approximated by applying the distance transform on a binary mask. We further propose a post-process technique to refine the background mask for more accurate results. We have tested the proposed method in the task of foreground segmentation. The experimental results indicate that the models trained using the dataset augmented with the new synthetics are more robust to illumination changes and are able to handle even intense lighting variations. As future work, more shapes can be explored, not only geometrical but also of arbitrary shapes, for representing shadows more realistically.

Chapter 5

Illumination-aware Multi-task GANs for Foreground Segmentation

In the previous chapter, we showed that augmenting the dataset by artificially changing the illumination of the input increases the generalisation abilities of the models. In this chapter, we make this process learnable. We develop a CNN architecture comprised of three GANs that decomposes and reconstructs the illumination of the scene, while performing foreground segmentation simultaneously in an end-to-end manner. The proposed architecture allows the model to effectively capture the semantic relationship between dark and bright images. By jointly optimising the GAN loss and the segmentation loss, our network simultaneously learns both tasks that mutually benefit each other. Furthermore, fusing features of images with varying illumination into the segmentation branch vastly improves the performance of the network. Comparative evaluations on highly challenging real and synthetic benchmark datasets (ESI [135], SABS [15]) demonstrate the robustness of TMT-GAN and its superiority over state-of-the-art approaches.

5.1 Introduction

With the recent success of Deep Learning in image segmentation [47, 21], high accuracy in BGS can be achieved in controlled environments, which can be either videos with minimal change in the background or images with adequate illumination and high contrast. However, it is a much harder problem in real-world scenarios, in which the illumination changes in the scene may cast shadows, cause reflection and even alter the color of objects. In unexpected, but not uncommon, scenarios such as a street light being suddenly switched off at night, the effect can be dramatic and unmanageable by existing models (Figure 5.6 and 5.8).

In this chapter, we tackle the aforementioned problems by proposing a Triple Multi-Task Generative Adversarial Network (TMT-GAN) for foreground segmentation; comprised of three separate GANs, each solving a different task. A naive approach would suggest using a single GAN to normalize the illumination of the input image and then perform foreground segmentation in a two-step manner. However, in this case the segmentation accuracy would be very sensitive to the reconstruction abilities of the GAN and would fail completely if the generated image is even slightly inaccurate. Our proposed TMT-GAN is specifically designed to solve two problems at the same time in a multi-task, end-to-end manner: decoding the illumination of the scene and performing bforeground segmentation. This is accomplished by generating a pair of low/high brightness images and using GANs to reconstruct each of them with the brightness level of the other. The foreground segmentation is then performed using multi-scale features extracted from different layers of the generators. The result is a unified system for robust BGS that addresses the weaknesses of existing approaches in videos featuring drastic illumination changes. Experimental results indicate the robustness of our proposed framework on benchmark datasets with significant change in illumination that outperforms state-of-the-art approaches.

The main contributions of this study are summarized as follows:

- We propose a novel end-to-end architecture based on a triple multi-task generative adversarial network (TMT-GAN) for background-foreground segmentation on videos with significant changes in illumination.
- We construct the supervision of the generators in a manner that increases the contrast between foreground and background and facilitates illumination-aware BGS. We jointly optimise the GAN loss and the segmentation loss to obtain optimal results.
- To the best of our knowledge, this is the first multi-task GAN with inputs of different degrees of brightness. We show that fusing features of images with varying illumination into the segmentation branch vastly improves the performance.

The rest of this chapter is structured as follows: Section 5.2 explains our methodology in detail and provides technical information. In Section 5.3, we introduce the datasets used in this study and present the experimental results. A summary and future work are discussed in Section 5.4.

5.2 Methodology

In this section, we introduce our proposed BGS framework and the proposed architecture, which is illustrated in Fig. 5.1. Given a video sequence, the proposed framework takes each individual frame as the input. Each image is first edited by increasing and decreasing the gamma value to create a pair of images with extreme illuminations (Section 5.2.1). Then, each edited image is fed into the VGG16 [115] network for extracting the deep features. Next, our framework learns a robust representation between the paired images. Motivated by the success of double GAN for image-to-image translation between different domains [168, 52, 152], we employ it for illumination decomposition by learning the differences between exceedingly bright/dark images. This is accomplished by reconstructing an image

of one domain with characteristics of the other. Furthermore, we extend these methods by appending an extra GAN for binary segmentation on the classes of foreground/background.

Overall, we use one encoder E_1 for general feature extraction and three generators G_b, G_d, G_s along with three discriminators D_b, D_d, D_s for the domain of bright images, dark images and binary segmentation respectively. Skip connections between layers are employed to all generators to aid the preservation of high-level information and edge alignment. In more detail, the output of three layers of VGG16 is extracted and concatenated with the corresponding features of the generators G_b and G_d which are of the same resolution. Basically, we divide our approach into three distinct parts: pre-processing, feature extraction and finally foreground segmentation. Each part is discussed in the following subsections.

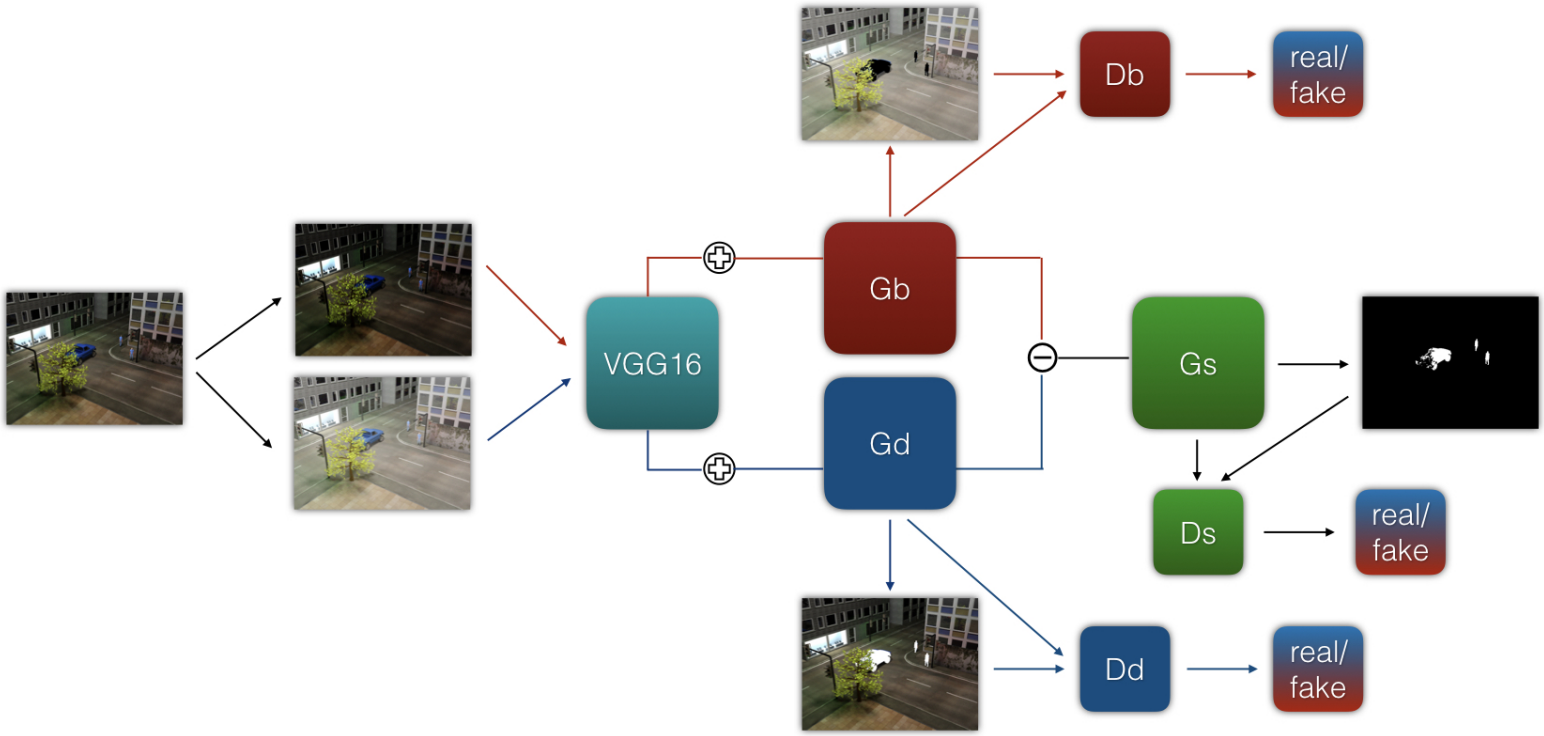


Fig. 5.1 The architecture of TMT-GAN. Given an image, a low/high brightness image pair is generated with the gamma function. Both images individually undergo feature extraction by $VGG16$, followed by image generation by G_b and G_d. Red arrows show the path for bright images and blue arrows for dark images. Multi-scale features are extracted from three inner layers of G_b and G_d and incorporated into the foreground segmenter G_s with pixel-wise subtraction. D_b, D_d and D_s are the discriminators of the respective generators and ensure the distribution matching between the real and generated images. The plus and minus signs indicate feature concatenation and element-wise subtraction respectively. The ground truth construction is explained in section 5.2.2.

5.2.1 Pre-processing with gamma correction

To ensure the robustness of our model against illumination changes, we create multi-scale inputs in regard to luminance. Specifically, given an image, we alter its brightness using gamma correction [100]. According to this approach, the intensity value of every pixel p is first normalised to the range [0,1] and then raised to the power of γ:

$$p_i^{\text{out}} = (p_i^{\text{in}}/255)^{\gamma} \quad i \in 1,\ldots,N, \tag{5.1}$$

where p^{out} and p^{in} are the pixels of the output and input image respectively and N is the total number of pixels. By setting $\gamma > 1$, the image becomes darker. Conversely, for $\gamma < 1$, the brightness is increased. As the γ value diverges from 1, the phenomenon becomes more extreme. Therefore, it is possible to generate an exceptionally bright/dark pair $\{I_b, I_d\}$ for each image regardless of its original brightness. The optimal value of γ is calculated adaptively and according to the average pixel intensity of the input image:

- $\gamma_d = 2.5$, $\gamma_b = 0.7$, if $\hat{p} \in \{0,\ldots,95\}$
- $\gamma_d = 2.0$, $\gamma_b = 0.5$, if $\hat{p} \in \{96,\ldots,120\}$
- $\gamma_d = 1.4$, $\gamma_b = 0.3$, if $\hat{p} \in \{121,\ldots,150\}$
- $\gamma_d = 1.0$, $\gamma_b = 0.1$, if $\hat{p} \in \{151,\ldots,255\}$

where $\hat{p}$ denotes the mean intensity values of the input image's pixels and γ_b, γ_d are the values of γ applied to the original input image for generating the input pair $\{I_b, I_d\}$.

5.2.2 Feature extraction

Transfer learning

As reported in recent work such as [71, 156], using a pre-trained CNN leads to an increased accuracy since it helps the model to converge to a better local minimum. In the proposed framework, and for fair comparison against the state-of-the-art, we use the pre-trained VGG16 [115] as the encoder. The same network is used as backbone in both OSVOS [16] and FgSegNet [71], while CascadeCNN [141] did not use a pre-trained network but developed a custom encoder. In addition, since it is only used for feature extraction, we only keep the first four blocks of VGG16 and discard the rest as in previous work [71].

Ground truth construction

Once the input pair is obtained, each image is individually fed to VGG16 for general feature extraction. Consequently, each set of features forms the input of the corresponding generator, which is of an identical structure. The ultimate goal of the generators is to learn the illumination of the image by altering its brightness, not unlike transforming an image from day to night [53] and vice versa. At the same time, the generators need to focus on the foreground and separate it from the background. This can be divided into two objectives:

- Brightness alteration to the extremes
- Foreground object identification

We can simultaneously optimise both tasks with a single loss function by constructing the supervision as follows: the illumination decomposition is supervised by altering the intensity of the pixel values as described in section 5.2.1. We can then train the network to detect the foreground objects by creating a large contrast between the foreground and the background pixels. To this end, the foreground pixels are assigned the value of $f_p = 255$ for dark

supervision images. Conversely, for bright images, we set $f_p = 0$. Since the predicted pixel values range from $[0, 255]$, there is a high contrast between the foreground and background objects. The process is supervised by the corresponding discriminators, which ensure that the distribution of the generated images matches that of the input images.

5.2.3 Foreground Fegmentation

Multi-scale feature fusion

Essentially, the deep features of G_b and G_d encode both illumination and saliency information. Moreover, they project the foreground object to two opposing extremities of the RGB spectrum. Therefore, the selection of the appropriate fusion mechanism now becomes apparent: subtraction. More specifically, we extract features of different resolutions from three layers of G_b and G_d, namely, G_{bi} and G_{di} respectively, where $i \in \{2, 4, 8\}$ denotes the downsampling ratio. To obtain the final features, we perform element-wise subtraction and scaling by applying the hyperbolic tangent function: $G_{si} = \tanh(G_{di} - G_{bi})$. Using *tanh* ensures that features of all layers will be on the same scale before merging, and also adds another layer of non-linearity. Finally, the foreground segmentation generator G_s accepts as input the features G_{si} and provides the final segmentation mask.

Attention

Attention-based CNNs have gathered intense interest among researchers recently [146, 147]. Intuitively, the attention mechanism is used for teaching the model to focus on specific parts of the input. Due to this feature, such a module is directly relevant to foreground segmentation, as it can potentially assist the model to focus on the foreground.

In the task of image segmentation, visual attention can be categorised into two parts: soft attention and hard attention. While the implementation varies wildly, generally hard

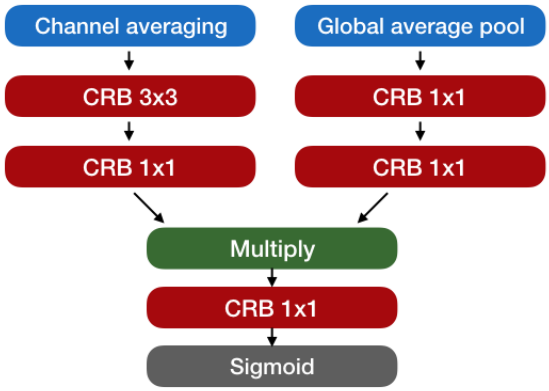


Fig. 5.2 The attention module

attention samples one region of the image at a time and is not differentiable; on the other hand, soft attention is, as it creates a probability map which is used to assign different weights to each pixel according to its significance on the task [120]. In addition, attention can be both supervised [76] and unsupervised [133].

We employ a self-supervised, soft attention mechanism. Inspired by Li et al. [70], we divide soft attention into spatial and channel attention, each of which is modelled with a separate stream of a similar structure. In particular, each stream consists of an average pooling layer and two convolutional layers. The average pooling layer is used to compress information across the feature maps, thereby generating a single-channel map with the most consistent activations. The first convolutional layer is adding the attention, while the second one is used for scaling. The two attention streams are fused with tensor multiplication. The attention module is depicted in Figure 5.2.

Three attention modules are embedded into the segmentation generator G_s. Each module operates in different resolution and accepts as input the subtracted features of the corresponding layers of G_b and G_d, after being convolved with a 5x5 filter and concatenated with the output of the previous layer. Therefore, they utilise information from different sources and resolutions to provide the refined features.

Foreground Segmentation Discriminator

To further increase the segmentation accuracy of the model, we append a discriminator, D_s, to the output of G_s. Basically, D_s discriminates between the generated mask of G_s and ground truth. In most cases, the foreground mask consists of a small number of objects of similarly defined boundaries and is easily discernible to noise. Therefore, D_s can lead G_s to generate higher quality segmentation masks in two ways: firstly, by reducing false-positive noise in the background and secondly, by ensuring that the foreground blobs are smooth and consistent without false-negative areas in their interior.

5.2.4 Training

To optimise the task of domain translation, we use a loss that combines the optimisation of the generators for image reconstruction and the discriminators for ensuring the generated image is as natural as possible:

$$\begin{aligned} L_t = & LD_d(G_d(x_d), t_d) + \alpha \left\| G_d(x_d) - t_d \right\| \\ & + LD_b(G_b(x_b), t_b) + \alpha \left\| G_b(x_b) - t_b \right\|, \end{aligned} \tag{5.2}$$

where x_b, x_d and t_b, t_d are the input image and the supervision of bright and dark images respectively, α is a hyper-parameter and LD stands for the cross-entropy loss of the discriminators. In our experiments, we set $\alpha = 20$. The first pair of terms trains the dark image GAN, while the second one trains the bright image GAN.

In the task of foreground segmentation, the classes of foreground and background are usually heavily imbalanced. To address this issue, we use the weighted cross-entropy loss, which is formally defined as follows:

$$G_s = wt[-\log \sigma(x)] + (1-t)[-\log (1-\sigma(x))], \tag{5.3}$$

where w is the weight coefficient, x is the predicted label, t is the target label and $\sigma(x) = \frac{1}{1+e^{-x}}$ is a sigmoid function. To punish false negatives in the loss function and balance the two classes, we set $w = 5$ by trial and error.

5.3 Results

To evaluate the effectiveness of our proposed method, we compare our method with the following state-of-the-art approaches:

- OSVOS [16], the creators of the Davis dataset [96] for video segmentation,
- FgSegNet [71], the best performer on the benchmark dataset CDnet2014 [43], and
- CascadeCNN [141], the third best performer on CDnet2014 and the second best with source code open to the public

on two challenging datasets with a strong focus on intense illumination changes to demonstrate the robustness of our method. In particular, the Stuttgart Artificial Background Subtraction dataset (SABS) [15] and ESI [134] datasets are used.

To ensure a fair comparison between all models, we use the same training hyper-parameters. In more detail, we set *batch size* $= 1$ (sample-by-sample update) to ensure the model fits in the GPU and *training epochs* $= 15$. When obtaining the binary segmentation map, we select the threshold that maximises the F-Measure. We also use pre-trained models to initialise the model parameters when applicable. Finally, all models are trained with the same training/testing split. We report the means and standard deviations of 5 runs.

5.3.1 Evaluation Metrics

We evaluate the models using a very wide variety of metrics which are commonly used in the task of BGS [43]: *Recall, Specificity, Precision, False Positive Rate, False Negative Rate,*

Percentage of Wrong Classifications, F-Measure and *Intersection over Union*. The scientific formulation of these metrics are as defined in equations 3.4-3.10 and 4.11, 4.12.

5.3.2 Implementation details

In the experiments, our proposed framework is implemented in Tensorflow with a single NVIDIA GeForce GTX 1060 GPU. Training was completed in approximately 6 hours. In terms of data pre-processing, all images are resized to 240x320 and normalised to $[-1,1]$. Shuffling is also applied, however there is no data augmentation employed. This is part of our effort to minimise the effect of other factors and ensure the fair comparison between the models.

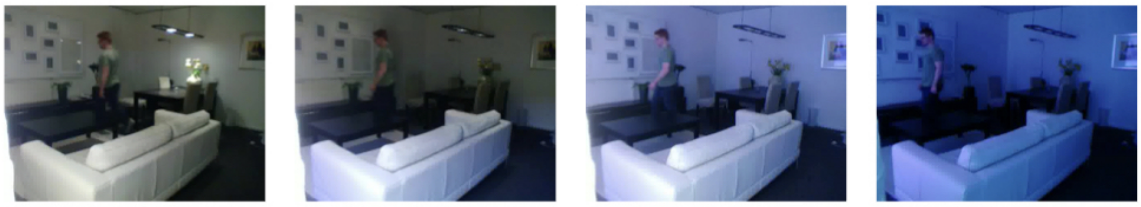
As mentioned before, we employ transfer learning by using the first four blocks of the pre-trained network VGG16 [115]. Computational efficiency is achieved with minimal loss of accuracy by freezing the first two blocks and only training the rest. Dropout is employed at the last block with keep probability $p_k = 0.5$.

5.3.3 SABS Dataset

The first dataset that is used for evaluating the experiments in this chapter is the SABS dataset [15], which was introduced in Section 4.3.2. However, the training set in the experiments of this chapter includes the *No Foreground Night* video as well. This sequence features only background movements and mild illumination changes like traffic lights and reflections. We find that extending the dataset with this video, which consists of 801 frames, improves the segmentation results of the models. As in chapter 4, the *Light Switch* video is used in the comparison as the testing data, since it is the most challenging video with sudden illumination changes. An example of 3 consecutive frames is illustrated in Figure 5.3b. Explicit details of the training/testing data split are stated in Table 5.1.

Scene	Frame indices
Training	
Darkening	1-800 (whole video)
No Foreground Night	1-801 (whole video)
Testing	
Light Switch	1-600 (whole video)

Table 5.1 Scenes and frame indices used in training and testing on the SABS dataset



(a) *Walking* video of ESI

(b) *Light Switch* video of SABS

Fig. 5.3 Consecutive frames in the SABS and ESI datasets featuring sudden illumination changes

The F-Measure indicates the average accuracy of the BGS. While the OSVOS [16], FgSegNet [71] and CascadeCNN [141] are having similar performance, our proposed method significantly outperforms the existing methods and achieved a much higher F-Measure value, as demonstrated in Table 5.4 and Figures 5.4, 5.5. This highlights the effectiveness of our method.

To evaluate the performance qualitatively, some examples of the BGS results are illustrated in Figure 5.6. Three different scenarios are presented in Figure 5.6, namely *normal* (row 1 and 4), *occlusion* (row 2) and *light off* (row 3). *Normal* scenes are having normal illumination in which the scene is bright in general and BGS can be done more easily. In *occlusion* scenes, some foreground objects are occluded by static objects which make the segmentation task more difficult. *Light off* scenes are those with the lights being switched off and results in significant illumination change over consecutive frames, which amounts to a very challenging situation.

From the results, it is demonstrated that the foreground masks (coloured in white) obtained using our method (Figure 5.6, rightmost column) are less noisy than those obtained using OSVOS [16], FgSegNet [71] and CascadeCNN [141]. Also, our results are closest to the ground truth in this test, which aligns well with the quantitative results presented in Table 5.4. In particular, our method significantly outperformed others in the more challenging (i.e. *occlusion* and *light off*) scenes (Figure 5.6 2nd and 3rd rows). Even in the normal scenario, the superiority of our method is evidenced by the well-defined boundaries, such as the light post being correctly classified as background in the first row and the shape of the wheels in the last row of Figure 5.6. On the other hand, OSVOS [16] had trouble adjusting to dynamic environments, as it incorrectly classified many pixels of the tree as foreground. For FgSegNet [71] and CascadeCNN [141] similar results were obtained, as they failed to accurately segment both of the cars, including the non-occluded car. In the *normal* scenes (1st and 4th rows of Figure 5.6), all methods performed well, and CascadeCNN [141] achieved

comparable performance with our method. However, CascadeCNN [141] tends to create masks with blurry edges/boundaries as depicted by the shape of the wheels in Figure 5.6.

In addition to the state-of-the-art approaches, we also compared our method with other existing methods and the results (F-measure) are illustrated in Figure 5.4 and 5.5. Again, our method outperformed the other methods. Note that the statistics are obtained from [114] and [55], and all the results are showing the performance (F-measure) on the same testing video sequence *Light Switch*.

Comparison against chapter 2

While TMT-GAN benefited from training with an extra video sequence (*No Foreground Night*), the chapter 2 model did not. Therefore, we directly compare against the best reported model, GL_{refine} or GL_r for short. We include GL_r in our comparison against the state-of-the-art models in Table 5.4.

As evidenced by the quantitative results in the previous Chapter, GL_r offered massive improvements over traditional models augmented with common techniques (Table 4.2). The effectiveness of our proposed data augmentation method is clearly demonstrated in Table 5.4, as GL_r surpasses the state-of-the-art in terms of balanced metrics like F-Measure and IoU despite having a simpler architecture. However, learning the illumination of a scene is even more effective as it brings an extra 7% improvement over manual brightness adjustments in terms of F-Measure score.

This improvement is also reflected in the qualitative evaluation. As demonstrated in Figures 4.11 and 5.6, the segmentation results of TMT-GAN feature better object boundaries and much less noise. The amount of false positives is kept to a minimum as the model inherently learns the scene illumination, as opposed to only learn from synthetics.

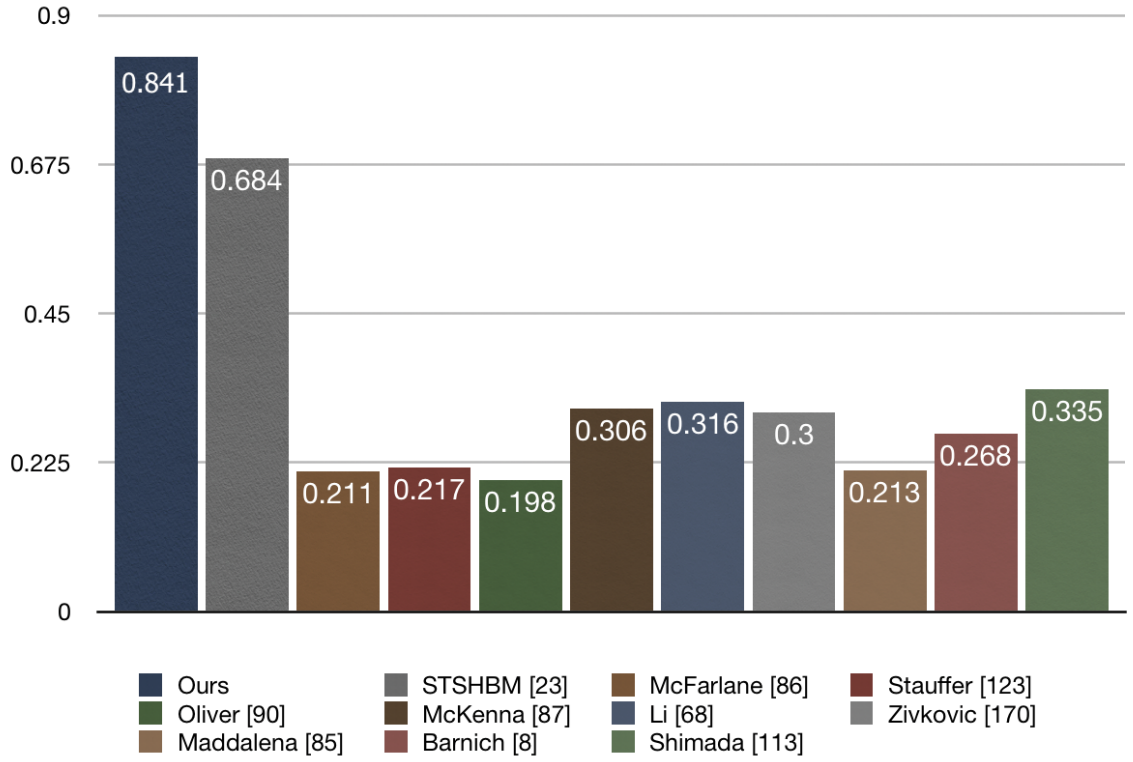


Fig. 5.4 Comparison of F-Measure values with state-of-the-art models on the *Light Switch* sequence of SABS. Statistics are taken from Shimada and Taniguchi [114].

We also compare against the failure cases of Chapter 4. The results of the proposed model of this chapter can be seen at Figure 5.7. Although some errors persist, there is a clear improvement compared to the model developed in the previous Chapter.

5.3.4 ESI Dataset

We further evaluate our method by the ESI [134] dataset which contains 8 video sequences filmed indoor, 3 of which being background-only. Throughout all the videos, various sources of light are being switched on and off which causes drastic changes to the illumination of the room. Examples of these changes are shown in Figure 5.3a. The data used for training and testing the models is listed in Table 5.2. The split between training and testing sets is specifically performed in a way that it separates the two without allowing the models to take a

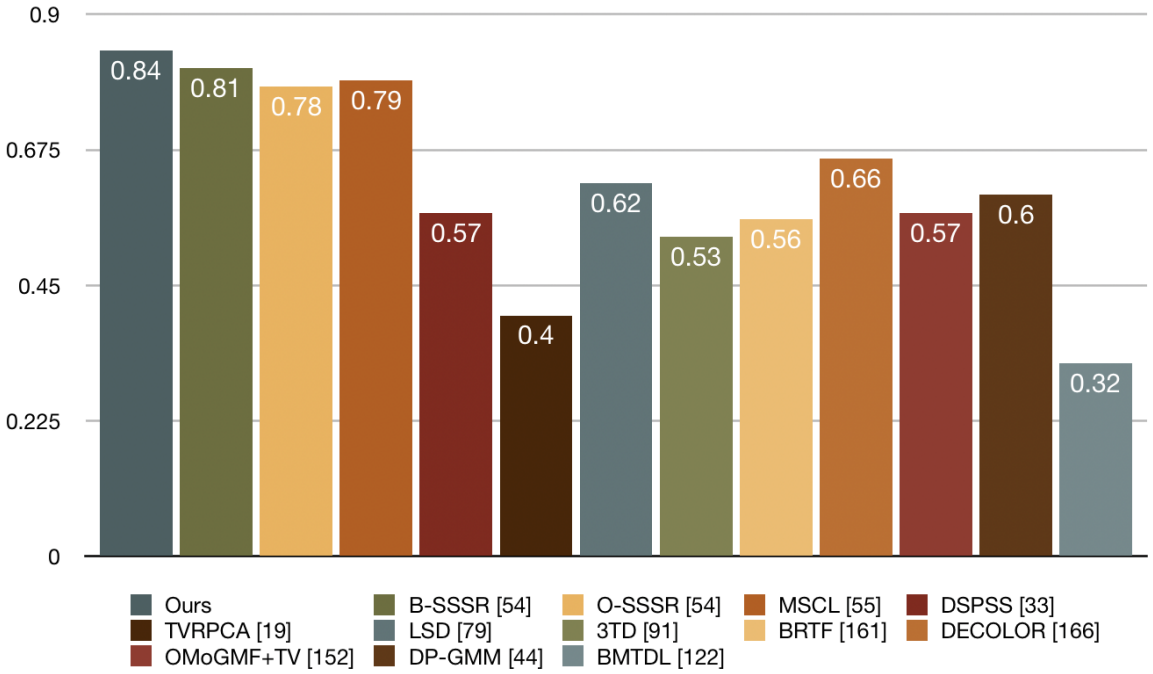


Fig. 5.5 Comparison of F-Measure values with state-of-the-art models on the *Light Switch* sequence of SABS. Statistics are taken from Javed et al. [55].

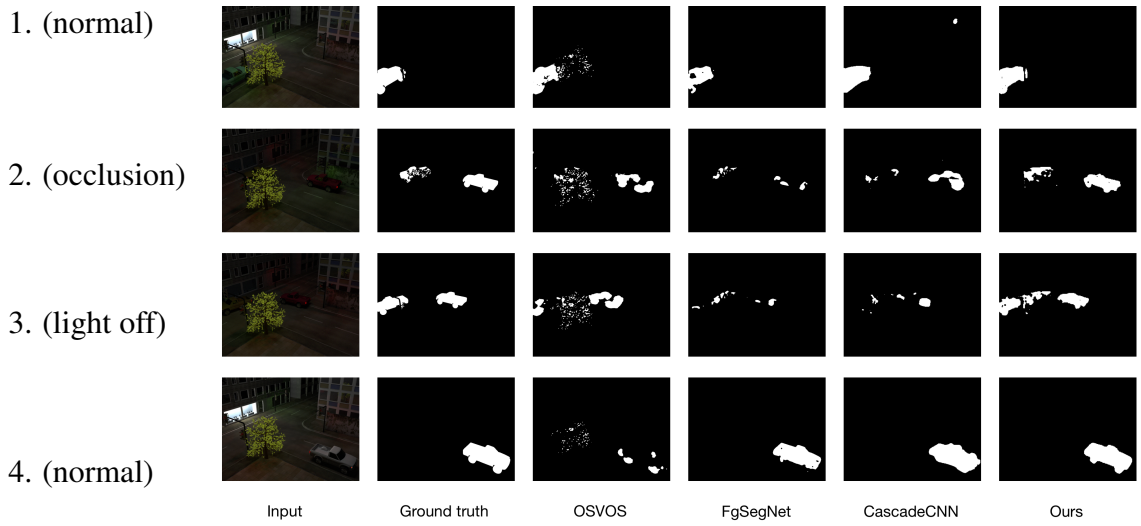


Fig. 5.6 Qualitative results on the SABS dataset

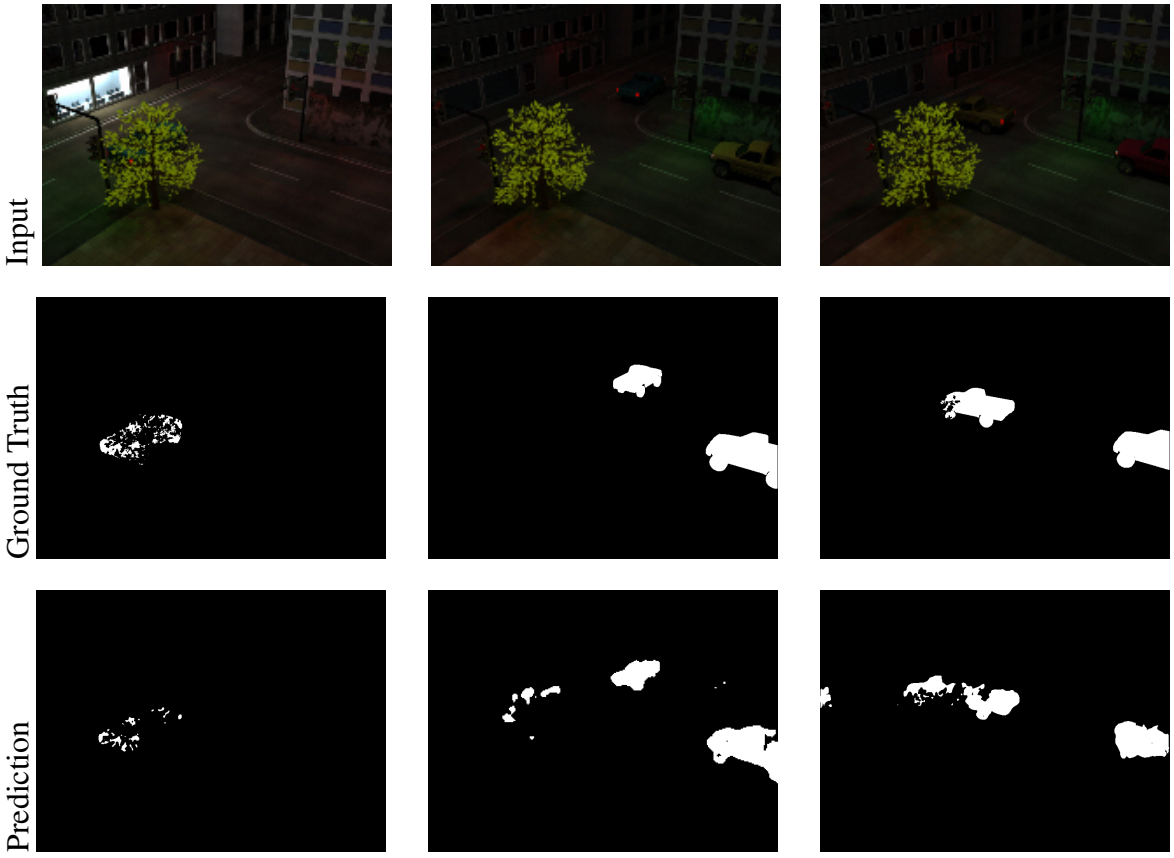


Fig. 5.7 Failure cases

Scene	Frame indices
Training	
Background scene 1	1-1375
Background scene 2	1-1093
house	1-401
walking	24-757
scene1	599-1238
scene2	1768-1836
Testing	
chair	90-663
scene1	489-589
scene2	1846-1921
sofa	34-418

Table 5.2 Scenes and frame indices used in training and testing on the ESI dataset

glimpse into the future. Instead of manually selecting representative frames from each video sequence as in [141, 71], we either select different videos for training and testing or split a video into two continuous parts, depending on whether they share the same background. Therefore, *scene1* and *scene2* are divided into two parts consisted of consecutive frames, the first being reserved for training and the latter for testing, since some foreground of these scenes has to be included in the training data. Nevertheless, the testing sequences are unseen data as indicated by the frame indices for *scene1* and *scene2* in Table 5.2 and the movement of the person (foreground) is completely different between the training and testing scenes. On the other hand, *chair* and *sofa* can be used for testing in their entirety, since they share the same background with *walking*. The results are presented in Table 5.6 and 5.8.

Model	Recall	Specificity	FPR	FNR	PWC
OSVOS	0.6121 ±0.0161	0.9922 ±0.00022	0.0107 ±0.00025	0.367 ±0.00243	1.7175 ±0.0061
FgSegNet	0.5618 ±0.0127	0.9963 ±0.00015	0.0029 ±0.00032	0.4344 ±0.00313	1.3278 ±0.0092
CascadeCNN	0.5781 ±0.0195	0.9961 ±0.00027	0.0055 ±0.00017	0.4119 ±0.00235	1.4756 ±0.0112
GL_{refine}	0.816 ±0.0142	0.9939 ±0.00018	0.0067 ±0.00017	0.2031 ±0.00288	1.1085 ±0.0088
TMT-GAN	0.8786 ±0.0122	0.9952 ±0.00017	0.0049 ±0.00014	0.1189 ±0.0021	0.7938 ±0.0045

Table 5.3 Results on the SABS (Light Switch) dataset. Best scores are highlighted in green colour, while second best are depicted in red font.

Model	FM	Precision	IoU
OSVOS	0.6187 ±0.0098	0.6257 ±0.0192	0.4456 ±0.0137
FgSegNet	0.6728 ±0.0084	0.8048 ±0.0163	0.4981 ±0.0159
CascadeCNN	0.637 ±0.0097	0.7324 ±0.0192	0.468 ±0.0178
GL_{refine}	0.7752 ±0.0074	0.7326 ±0.0184	0.6278 ±0.0145
TMT-GAN	0.8411 ±0.0063	0.8076 ±0.0124	0.7245 ±0.0109

Table 5.4 Results on the SABS (Light Switch) dataset. Best scores are highlighted in green colour, while second best are depicted in red font.

Method	**Category**			
	chair		*scene1*	
	F-measure	PWC	F-measure	PWC
OSVOS	0.7455 ±0.0079	1.4763 ±0.0074	0.8687 ±0.0061	3.415 ±0.00103
FgSegNet	0.8142 ±0.00117	0.9321 ±0.0064	0.8974 ±0.0071	2.281 ±0.0076
CascadeCNN	0.7285 ±0.0089	1.4754 ±0.0132	0.7613 ±0.0108	4.95 ±0.0096
TMT-GAN	0.8391 ±0.0056	0.838 ±0.0067	0.9101 ±0.0054	1.928 ±0.0071

Table 5.5 Results on the ESI dataset by category

Method	**Category**			
	scene2		*sofa*	
	F-measure	PWC	F-measure	PWC
OSVOS	0.8947 ±0.0077	1.8471 ±0.0065	0.6418 ±0.0073	4.69 ±0.0103
FgSegNet	0.9276 ±0.0119	1.4377 ±0.0068	0.711 ±0.0122	3.8721 ±0.0081
CascadeCNN	0.8991 ±0.0125	2.2339 ±0.0059	0.5979 ±0.0088	5.5719±0.0069
TMT-GAN	0.9381 ±0.0055	1.337 ±0.007	0.7625 ±0.0091	3.3192 ±0.0058

Table 5.6 Results on the ESI dataset by category

Model	Recall	Specificity	FPR	FNR	PWC
OSVOS	0.7729 ±0.0218	0.9903 ±0.00032	0.0104 ±0.00033	0.2278 ±0.00277	2.3041 ±0.0093
FgSegNet	0.8208 ±0.0203	0.9922 ±0.00021	0.0074 ±0.00019	0.1788 ±0.00224	1.7378±0.0074
CascadeCNN	0.7479±0.0223	0.9828 ±0.00034	0.0147 ±0.00041	0.2415 ±0.00268	2.887 ±0.0149
TMT-GAN	0.8986±0.0233	0.9923 ±0.00027	0.007 ±0.00016	0.0997 ±0.00281	1.2623 ±0.0087

Table 5.7 Results on the ESI dataset

Model	FM	Precision	IoU
OSVOS	0.7879 ±0.01	0.8027 ±0.0176	0.6479 ±0.0124
FgSegNet	0.8394 ±0.0045	0.8574 ±0.0225	0.7246 ±0.0162
CascadeCNN	0.7425 ±0.0069	0.7331 ±0.017	0.5905 ±0.0202
TMT-GAN	0.8881 ±0.0071	0.8781 ±0.0243	0.7988 ±0.0148

Table 5.8 Results on the ESI dataset

Removed Component	F-Measure	Recall	Precision
Attention	0.7978±0.0064	0.79245 ±0.0145	0.8112 ±0.0136
D_f	0.82891±0.0068	0.78601 ±0.0122	0.87245 ±0.0191
Weighted loss	0.73404±0.0089	0.64792 ±0.0245	0.82741 ±0.0278
None	**0.84429**±0.0088	0.8673 ±0.0238	0.82439 ±0.0369

Table 5.9 F-Measure values of the model trained on SABS if a component is removed

Removed Component	F-Measure	Recall	Precision
Attention	0.88421±0.0068	0.90343 ±0.0176	0.86427 ±0.0141
D_f	0.84793±0.0045	0.86349 ±0.0181	0.83401 ±0.0204
Weighted loss	0.83448±0.0076	0.81865 ±0.0163	0.86042 ±0.0166
None	**0.88845**±0.0064	0.90123±0.0104	0.87549 ±0.0122

Table 5.10 F-Measure values of the model trained on ESI if a component is removed

Again, the F-Measure indicates our method significantly outperforms the OSVOS [16], FgSegNet [71] and CascadeCNN [141]. This highlights the consistency and robustness of our method.

The qualitative results are illustrated in Figure 5.8. To clearly show the way each method handles sudden illumination changes, we provide the segmentation maps on consecutive frames of each video sequence of the testing set where the illumination of the scene changes drastically. Among the 4 videos, *Scene2* features the slightest change in illumination, which explains why all methods are performed well (Row 3-4 in Figure 5.8).

It can be seen that state-of-the-art models have considerable noise in low-light frames. FgSegNet [71] has very few false positives. However, it comes with a cost of a large number of false negatives. OSVOS [16] and CascadeCNN [141] on the other hand, provide better person silhouettes but also have many false positives. All in all, our method (Figure 5.8 rightmost column) achieves accurate segmentation maps even in images of low brightness with very minimal false positives and negatives, as seen in comparison to the ground truth.

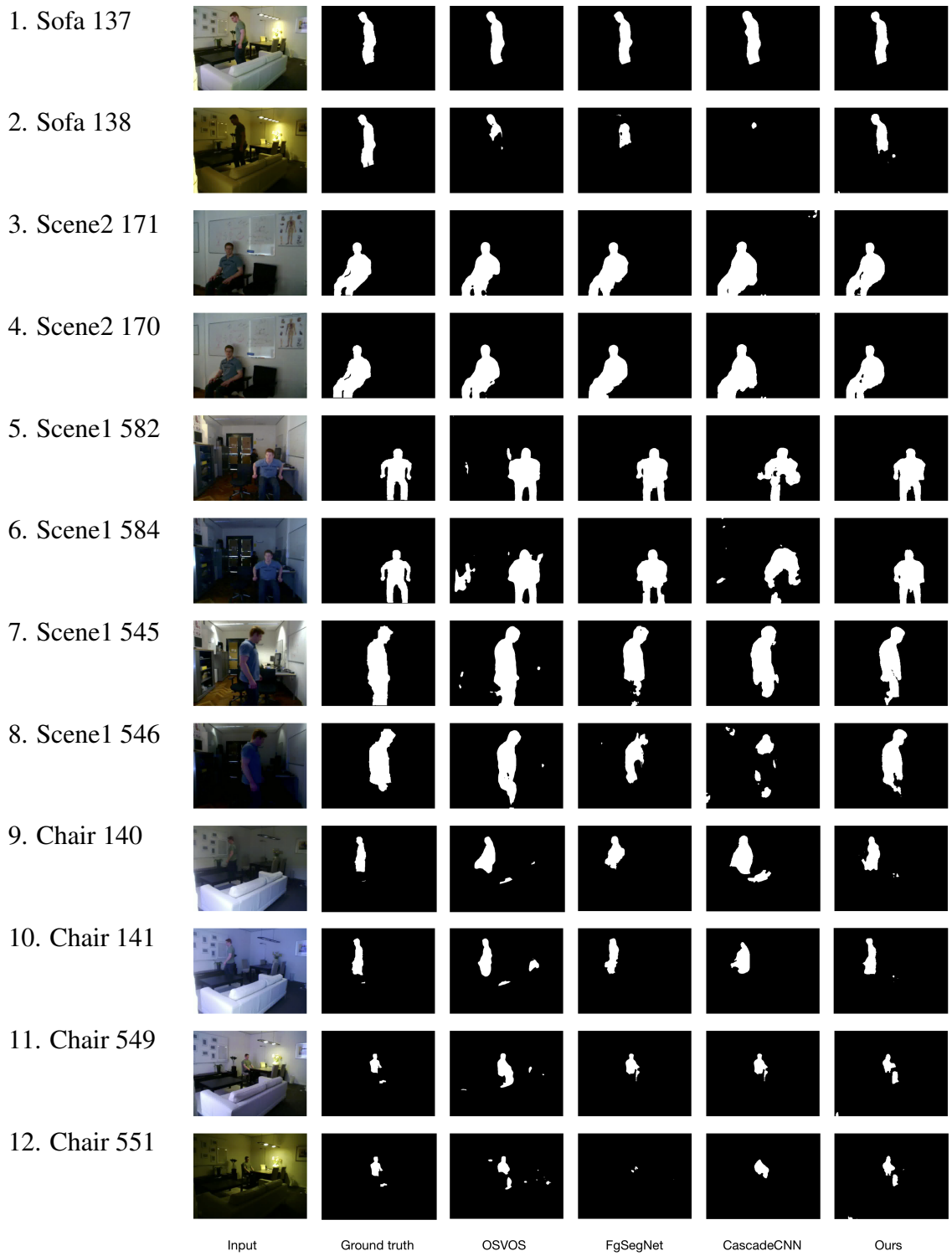


Fig. 5.8 Qualitative results on the ESI dataset

Complexity	OSVOS	FgSegNet	CascadeCNN	TMT-GAN
Parameters	14,917,637	8,155,713	157,537	83,283,620

Table 5.11 Analysis on the parameter count of the evaluated models

5.3.5 Complexity comparison

We compare the parameter count of the models at Table 5.11. CascadeCNN has an extremely small number of parameters, because it operates in patches. Specifically, the input size is only 31x31 pixels, which means that the input image has to be broken into multiple pieces and processed by the model many times. In addition, CascadeCNN employs multi-scale inputs, therefore each patch will be processed three times. On top of these, it is cascaded model, meaning that the output of the network is fed to another network (with the same weights) to be processed again. As a result, the processing time increases even more. Multi-scale inputs are employed by FgSegNet as well. OSVOS on the other hand, has to be pre-trained in a huge, generic dataset. Our model is larger in terms of overall number of parameters, however each image is processed by a single forward pass of the neural network and we only pre-train the feature extractor.

5.3.6 Ablation studies

In this subsection, we justify the decisions we made in the proposed framework by conducting a series of ablation tests. In particular, we evaluate the performance of the proposed model by testing the effect on removing individual components on foreground-background segmentation tasks. The results are shown in Table 5.10.

From the results, it can be seen that all modules improve the FM performance in both datasets. First of all, it is shown that removing the weighted loss function and training with regular cross-entropy significantly affects the performance in the SABS dataset, but has a lower impact on the ESI dataset. This is because there is a higher imbalance on the

foreground/background classes in the SABS dataset, as the foreground objects are -mostly- smaller in size. Similarly, the attention module has a larger contribution on SABS than ESI because it is generally a more challenging dataset with smaller foreground objects, thus the effect is more profound. Finally, adding a discriminator on the G_s module also has a beneficial effect, as it forces G_s to create better quality masks.

5.4 Conclusion

In this research, we have proposed a foreground segmentation method based on adversarial learning and feature fusion, in order to obtain robust segmentation results even under intense illumination changes. It consists of three GANs: two for illumination translation (dark to bright and vice versa), and one for foreground segmentation. Our model successfully addresses the problems of existing methods in the most challenging datasets by learning illumination-aware features and using them for segmentation in a multi-task manner. Extensive experimental results demonstrate the superiority of the proposed method against state-of-the-art approaches in both normal and challenging scenarios and show its robustness in handling sudden and gradual illumination changes.

Furthermore, we have demonstrated that attention is a useful mechanism that can improve results by guiding the model to attend at the most discriminative features, according to the current input. Therefore, illumination noise is kept to a minimum.

The same can be said for class weighting in the loss function, especially in datasets that feature very small objects. In contrast to image classification where it is possible to add/remove training samples to balance the classes, pixel-wise classification is inherently different and using a weighted loss function is crucial. Finally, supervising the segmentation task with a discriminator improves results even more.

As future work, we intend to replace VGG16 with a deeper and most recent architecture for better feature extraction. The segmentation network can be improved as well if a pre-trained network is used. Finally, a larger batch size and a learning rate decay with warm restarts can further improve results [83].

Chapter 6

Conclusion and Future Work

6.1 Summary

In this thesis, we have addressed the current limitations of existing systems in the task of foreground segmentation in videos. However, even though our focus remains on the aforementioned area, the research within this thesis covers many closely related areas such as image segmentation, feature extraction, image-to-image translation and others.

6.1.1 3D convolutions for spatio-temporal context

Most of the previous research on image segmentation in general, but also in background subtraction, consider a video as a mere collection of frames and disregard temporal continuity [141, 5]. While this approach works well in datasets with relatively static backgrounds, it becomes problematic when challenged in videos that feature substantial changes over time.

In Chapter 3, we proposed a deep learning model which operates in a window of the 10 most recent frames of a video and performs 3D convolutions to extract spatio-temporal features. As a result, the model is able to learn the changes between frames and adapt.

Experimental results demonstrate the effectiveness of the proposed system over the state-of-the-art.

6.1.2 Semantic data augmentation for adapting to illumination changes

Existing data augmentation methods are very useful for enlarging a small dataset and improve the generalisation of a deep learning model, which needs a large amount of training data to converge. Simple functions as horizontal/vertical rotation, crop and noise addition do well in creating new images, however this process offers no semantic meaning.

In Chapter 4 we presented a data augmentation method for generating not only an endless stream of images, but also an infinite combination of local and global illumination changes and applying them to each input image. We have shown that it is possible to create very realistic illumination changes by using the euclidean transform. We further proposed a post-processing method based on frame continuity for removing noise from the final segmentation map. The proposed method is extremely lightweight and as such, adds minimal computational overhead. Experimental results show that training baseline models with the augmented data leads to very significant improvements compared to traditional augmentation methods.

6.1.3 Multi-task GANs for learning illumination changes

In cases of videos with minimal background change, training a model with some representative samples is enough to achieve good accuracy [71]. However, in challenging datasets that feature significant illumination changes, the state-of-the-art models fail. This is especially evident in the "light switch" scenario, where the illumination changes instantly.

We extended Chapter 4 and proposed a deep learning model which learns to alter the illumination of the scene and perform foreground segmentation end-to-end. We have demonstrated that this method is far superior than manual augmentation and we have obtained

further improvements compared to Chapter 4. We have also compared with and surpassed three state-of-the-art models in two datasets, as well as with a plethora of non - deep learning models. Ablation studies were conducted to justify the design choices of the model architecture.

6.2 Review of Contributions

The contributions of this thesis are summarized as follows:

- We proposed a 3D CNN for foreground segmentation, which is able to model spatial and temporal changes simultaneously. The model operates without the need of maintaining a separate background model and can handle several dozens of different videos.
- We proposed a novel image augmentation technique for robust foreground segmentation under illumination changes. The method generates new synthetic images by applying semantically sound image transformations based on local and global illumination effects. We further proposed a post-processing method for noise removal.
- We proposed a triple GAN for robust foreground segmentation, which is able to learn and alter the illumination of the scene and perform BGS in an end-to-end manner.

6.3 Directions for Future Work

In this Section we discuss the potential future work that can be done to extend the research presented in this thesis. First of all, the computational costs of the 3D CNN can be alleviated by optimising the length of its input window. Ideally, this could be done in an adaptive manner, so that the benefits from the trade-off between accuracy and efficiency will be

maximised. A promising way to realise this would be to quantify the difference between the current frame and the previous online. Depending on the outcome, past frames could be included as input in order to help the model overcome sudden changes in the scene. Since those changes are usually a rare occurrence, the computational savings can be very significant.

Certainly, another interesting way to increase the efficiency of the models presented in this thesis would be to replace the encoder with lightweight alternatives. This could be achieved with minimal, or none at all, loss in accuracy by optimal architecture search techniques like ENAS [97] and PNAS [75]. In this case however, pre-training the model in large datasets as COCO [73] or PASCAL VOC [37] would potentially lead to improvements in segmentation results. Alternatively, already trained, out-of-the-box pre-trained lightweight models can be used, such as MobileNet [50] or ShuffleNet [85]. To further reduce the model parameters, pruning methods can be employed [49].

With the future direction for improving the efficiency of our models discussed in last Section, we believe that the proposed frameworks could be applied for real-time applications without sacrifice in accuracy.

Finally, a fascinating and potentially fruitful research direction would be developing multi-task networks for domain adaptation and foreground segmentation. In Chapter 5 we showed that embedding the image-to-image translation task between different domains improves the robustness of the model by a large margin. In the future, we would like to research different approaches for domain adaptation and segmentation, which would not require to create multiple outputs. A primary example of such a system would use a discriminator at the deep feature space rather than at the decoder, as in [62]. As a result, the trained model would be highly robust across very different representations of the same scene.

References

[1] Akilan, T., Wu, Q. J., Safaei, A., Huo, J., and Yang, Y. (2019). A 3D CNN-LSTM-Based Image-to-Image Foreground Segmentation. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–13.

[2] Akilan, T., Wu, Q. J., and Yang, Y. (2018). Fusion-based foreground enhancement for background subtraction using multivariate multi-model gaussian distribution. *Information Sciences*, 430-431:414–431.

[3] Allebosch, G., Van Hamme, D., Deboeverie, F., Veelaert, P., and Philips, W. (2016). *C-EFIC: Color and Edge Based Foreground Background Segmentation with Interior Classification*, pages 433–454. Springer International Publishing, Cham.

[4] Anoosheh, A., Sattler, T., Timofte, R., Pollefeys, M., and Gool, L. V. (2018). Night-to-day image translation for retrieval-based localization. *arXiv:1809.09767v1*.

[5] Babaee, M., Dinh, D. T., and Rigoll, G. (2017). A deep convolutional neural network for background subtraction. *CoRR*, abs/1702.01731.

[6] Badrinarayanan, V., Kendall, A., and Cipolla, R. (2016). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:2481–2495.

[7] Bakkay, M. C., Rashwan, H. A., Salmane, H., Khoudour, L., Puigtt, D., and Ruichek, Y. (2018). Bscgan: Deep background subtraction with conditional generative adversarial networks. *2018 25th IEEE International Conference on Image Processing (ICIP)*.

[8] Barnich, O. and Van Droogenbroeck, M. (2011). ViBe: A universal background subtraction algorithm for video sequences. *IEEE Transactions on Image Processing*, 20(6):1709–1724.

[9] Bazzani, L., Cristani, M., and Murino, V. (2013). Symmetry-driven accumulation of local features for human characterization and re-identification. *Computer Vision and Image Understanding*, 117(2):130–144.

[10] Berjón, D., Cuevas, C., Morán, F., and García, N. (2018). Real-time nonparametric background subtraction with tracking-based foreground update. *Pattern Recognition*, 74:156–170.

[11] Bianco, S., Ciocca, G., and Schettini, R. (2015). How Far Can You Get By Combining Change Detection Algorithms? *IEEE Transactions on Image Processing*, abs/1505.0:1–10.

[12] Boulmerka, A. and Allili, M. S. (2018). Foreground segmentation in videos combining general gaussian mixture modeling and spatial information. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(6):1330–1345.

[13] Bouwmans, T. and Zahzah, E. H. (2014). Robust PCA via Principal Component Pursuit: A review for a comparative evaluation in video surveillance. *Computer Vision and Image Understanding*, 122:22–34.

[14] Braham, M. and Van Droogenbroeck, M. (2016). Deep background subtraction with scene-specific convolutional neural networks. *International Conference on Systems, Signals, and Image Processing*.

[15] Brutzer, S., Hoferlin, B., and Heidemann, G. (2011). Evaluation of background subtraction techniques for video surveillance. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1937–1944.

[16] Caelles, S., Maninis, K., Pont-Tuset, J., Leal-Taixé, L., Cremers, D., and Gool, L. V. (2017). One-shot video object segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5320–5329.

[17] Cai, Z., Fan, Q., Feris, R. S., and Vasconcelos, N. (2016). A unified multi-scale deep convolutional neural network for fast object detection. In *Computer Vision – ECCV 2016*, pages 354–370, Cham. Springer International Publishing.

[18] Candès, E. J., Li, X., Ma, Y., and Wring, J. (2011). Robust principal component analysis? *Journal of the Association for Computing Machinery*, 53(3):3179–213.

[19] Cao, X., Yang, L., and Guo, X. (2016). Total variation regularized rpca for irregularly moving object detection under dynamic background. *IEEE Transactions on Cybernetics*, 46(4):1014–1027.

[20] Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2015a). Semantic image segmentation with deep convolutional nets and fully connected crfs. *ICLR*, abs/1412.7062.

[21] Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2016a). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915.

[22] Chen, L.-C., Yang, Y., Wang, J., Xu, W., and Yuille, A. L. (2016b). Attention to scale: Scale-aware semantic image segmentation. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3640–3649.

[23] Chen, M., Wei, X., Yang, Q., Li, Q., Wang, G., and Yang, M.-H. (2018). Spatiotemporal gmm for background subtraction with superpixel hierarchy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6):1518–1525.

[24] Chen, Y., Wang, J., and Lu, H. (2015b). Learning sharable models for robust background subtraction. In *Proceedings - IEEE International Conference on Multimedia and Expo*, volume 2015-August.

[25] Chen, Y., Wang, J., Zhu, B., Tang, M., and Lu, H. (2017). Pixel-wise deep sequence learning for moving object detection. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1.

[26] Cheung, S. C. S. and Kamath, C. (2005). Robust background subtraction with foreground validation for urban traffic video. *Eurasip Journal on Applied Signal Processing*, 2005(14):2330–2340.

[27] Chollet, F. et al. (2015). Keras. https://keras.io.

[28] Cinelli, L. P., Thomaz, L. A., da Silva, A. F., da Silva, E. A., and Netto, S. L. (2017). Foreground segmentation for anomaly detection in surveillance videos using deep residual networks. In *Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBRT)*.

[29] Conneau, A., Schwenk, H., Barrault, L., and Lecun, Y. (2016). Very deep convolutional networks for natural language processing. *KI - Künstliche Intelligenz.*, 26.

[30] Cuevas, C., Yáñez, E. M., and García, N. (2016). Labeled dataset for integral evaluation of moving object detection algorithms: Lasiesta. *Computer Vision and Image Understanding*, 152:103–117.

[31] De Gregorio, M. and Giordano, M. (2017). CwisarDH$^+$: Background detection in rgbd videos by learning of weightless neural networks. In *New Trends in Image Analysis and Processing – ICIAP 2017*, pages 242–253, Cham. Springer International Publishing.

[32] Dongdong Zeng, Ming Zhu, A. K. (2019). Combining background subtraction algorithms with convolutional neural network. *Journal of Electronic Imaging*, 28(1):1 – 6.

[33] Dumoulin, V. and Visin, F. (2016). A guide to convolution arithmetic for deep learning. *CoRR, abs/1603.07285*.

[34] Ebadi, S. E. and Izquierdo, E. (2018). Foreground Segmentation with Tree-Structured Sparse RPCA. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40:2273–2280.

[35] Eigen, D., Krishnan, D., and Fergus, R. (2013). Restoring an image taken through a window covered with dirt or rain. In *Proceedings of the 2013 IEEE International Conference on Computer Vision*, ICCV '13, pages 633–640, Washington, DC, USA. IEEE Computer Society.

[36] Elgammal, A., Harwood, D., and Davis, L. (2000). Non-parametric model for background subtraction. *Proc. ECCV*, 1843:751–767.

[37] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338.

[38] Friedman, N. and Russell, S. (1997). Image Segmentation in Video Sequences: A Probabilistic Approach. *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 175–181.

[39] Gao, Y., Cai, H., Zhang, X., Lan, L., and Luo, Z. (2018). Background subtraction via 3d convolutional neural networks. *2018 24th International Conference on Pattern Recognition (ICPR)*.

[40] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

[41] Goyette, N., Jodoin, P. M., Porikli, F., Konrad, J., and Ishwar, P. (2012). changedetection.net: A new change detection benchmark dataset. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8.

[42] Goyette, N., Jodoin, P.-M., Porikli, F., Konrad, J. and Ishwar, P. (2017). Change detection benchmark website. http://jacarini.dinf.usherbrooke.ca/results2014/. Accessed: 2017-07-30.

[43] Goyette, N., Jodoin, P.-M., Porikli, F., Konrad, J. and Ishwar, P. (2018). Change detection benchmark website. http://jacarini.dinf.usherbrooke.ca/results2014/. Accessed: 2018-10-30.

[44] Guyon, C., Bouwmans, T., and Zahzah, E.-H. (2013). Foreground Detection via Robust Low Rank Matrix Decomposition Including Spatio-Temporal Constraint. *Computer Vision - ACCV 2012 Workshops*, pages 315–320.

[45] Haines, T. S. and Xiang, T. (2014). Background Subtraction with Dirichlet Process Mixture Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(4):670–683.

[46] Han, B. (2007). Real-time subspace-based background modeling using multi-channel data. *Advances in Visual Computing*, pages 162–172.

[47] He, K., Gkioxari, G., Dollár, P., and Girshick, R. B. (2017a). Mask r-cnn. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988.

[48] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

[49] He, Y., Zhang, X., and Sun, J. (2017b). Channel pruning for accelerating very deep neural networks. In *The IEEE International Conference on Computer Vision (ICCV)*.

[50] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861.

[51] Hu, Z., Turki, T., Phan, N., and Wang, J. T. L. (2018). A 3d atrous convolutional long short-term memory network for background subtraction. *IEEE Access*, 6:43450–43459.

[52] Huang, X., Liu, M.-Y., Belongie, S. J., and Kautz, J. (2018). Multimodal unsupervised image-to-image translation. *CoRR*, abs/1804.04732.

[53] Isola, P., Zhu, J., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976.

[54] Jaderberg, M., Simonyan, K., Zisserman, A., and kavukcuoglu, k. (2015). Spatial transformer networks. In *Advances in Neural Information Processing Systems 28*, pages 2017–2025. Curran Associates, Inc.

[55] Javed, S., Mahmood, A., Al-Maadeed, S., Bouwmans, T., and Jung, S. K. (2018). Moving Object Detection in Complex Scene Using Spatiotemporal Structured-Sparse RPCA. *IEEE Transactions on Image Processing*, 28(2):1007–1022.

[56] Javed, S., Mahmood, A., Bouwmans, T., and Soon, K. (2017). Background-foreground modeling based on spatiotemporal sparse subspace clustering. *IEEE T-IP*.

[57] Jeeva, S. and Sivabalakrishnan, M. (2015). Survey on background modeling and foreground detection for real time video surveillance. In *Procedia Computer Science*, volume 50, pages 566–571.

[58] Ji, S., Yang, M., Yu, K., and Xu, W. (2013). 3D convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–31.

[59] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

[60] Jiang, S. and Lu, X. (2017). WeSamBE: A Weight-Sample-Based Method for Background Subtraction. *IEEE Transactions on Circuits and Systems for Video Technology*, PP(99):1–1.

[61] KaewTraKulPong, P. and Bowden, R. (2002). An improved adaptive background mixture model for real-time tracking with shadow detection. *Video-Based Surveillance Systems*, pages 135–144.

[62] Kamnitsas, K., Baumgartner, C., Ledig, C., Newcombe, V., Simpson, J., Kane, A., Menon, D., Nori, A., Criminisi, A., Rueckert, D., and Glocker, B. (2017). Unsupervised domain adaptation in brain lesion segmentation with adversarial networks. In *Information Processing in Medical Imaging*, pages 597–609, Cham. Springer International Publishing.

[63] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-Scale Video Classification with Convolutional Neural Networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732.

[64] Kim, W. and Jung, C. (2017). Illumination-invariant background subtraction: Comparative review, models, and prospects. *IEEE Access 5*, pages 8369–384.

[65] Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv e-prints*, page arXiv:1412.6980.

[66] Kokkinos, I. (2016). Pushing the boundaries of boundary detection using deep learning. *ICLR*.

[67] Krizhevsky, A., Sutskever, I., and Geoffrey E., H. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems 25 (NIPS2012)*, pages 1–9.

[68] Li, C. and Ming, Y. (2019). Three-stream convolution networks after background subtraction for action recognition. *Artificial Intelligence and Soft Computing*, pages 12–24.

[69] Li, W. Huang, I. G. and Tian, Q. (2003). Foreground object detection from videos containing complex background. *Proc. 11th ACM Int. Conf. Multimedia*, pages 2–10.

[70] Li, W., Zhu, X., and Gong, S. (2018). Harmonious attention network for person re-identification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[71] Lim, L. A. and Keles, H. Y. (2018). Foreground segmentation using convolutional neural networks for multiscale feature encoding. *Pattern Recognition Letters*, 112:256 – 262.

[72] Lin, C., Yan, B., and Tan, W. (2018). Foreground detection in surveillance video with fully convolutional semantic network. *2018 25th IEEE International Conference on Image Processing (ICIP)*.

[73] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. *Computer Vision – ECCV 2014*, pages 740–755.

[74] Liu, C., Wu, X., and Shu, X. (2018a). Learning-based dequantization for image restoration against extremely poor illumination. *arXiv:1803.01532v2*.

[75] Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., Yuille, A., Huang, J., and Murphy, K. (2018b). Progressive neural architecture search. In *The European Conference on Computer Vision (ECCV)*.

[76] Liu, L., Utiyama, M., Finch, A., and Sumita, E. (2016a). Neural machine translation with supervised attention. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3093–3102. The COLING 2016 Organizing Committee.

[77] Liu, R., Lan, X., Yuen, P. C., and C Feng, G. (2016b). Robust visual tracking using dynamic feature weighting based on multiple dictionary learning. In *EUSIPCO*, pages 2166–2170.

[78] Liu, R., Lin, Z., Wei, S., and Su, Z. (2011). Solving Principal Component Pursuit in Linear Time via l1 Filtering. *arXiv preprint arXiv:1108.5359*.

[79] Liu, X., Yao, J., Hong, X., Huang, X., Zhou, Z., Qi, C., and Zhao, G. (2018c). Background subtraction using spatio-temporal group sparsity recovery. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(8):1737–1751.

[80] Liu, X., Zhao, G., Yao, J., and Qi, C. (2015a). Background subtraction based on low-rank and structured sparse decomposition. *IEEE T-IP*, 24(8):2502–2514.

[81] Liu, Z., Li, X., Luo, P., Loy, C. C., and Tang, X. (2015b). Semantic image segmentation via deep parsing network. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2015 International Conference on Computer Vision, ICCV 2015, pages 1377–1385.

[82] Long, J., Shelhamer, E., and Darrell, T. (2014). Fully Convolutional Networks for Semantic Segmentation. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440.

[83] Loshchilov, I. and Hutter, F. (2016). SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983.

[84] Lu, X. (2014). A multiscale spatio-temporal background model for motion detection. *2014 IEEE International Conference on Image Processing (ICIP)*.

[85] Ma, N., Zhang, X., Zheng, H.-T., and Sun, J. (2018). Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *The European Conference on Computer Vision (ECCV)*.

[86] Maddalena, L. and Petrosino, A. (2008). A self-organizing approach to background subtraction for visual surveillance applications. *IEEE Trans. Image Process.*, 17(7):1168–1177.

[87] McFarlane, N. J. B. and Schofield, C. P. (1995). Segmentation and tracking of piglets in images. *Machine Vision and Applications*, 8(3):187–193.

[88] McKenna, S. J., Jabri, S., Duric, Z., Rosenfeld, A., and Wechsler, H. (2000). Tracking groups of people. *Comput. Vis. Image Understanding*, 80(1):42–56.

[89] Mittal, A. and Paragios, N. (2004). Motion-based background subtraction using adaptive kernel density estimation. *Computer Vision and Pattern Recognition*, 2:302–309.

[90] Olah, Christopher (2015). Understanding lstm networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs//. Accessed: 2019-06-21.

[91] Oliver, N., Rosario, B., and Pentland, A. (2000). A bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):831–843.

[92] Oreifej, O., Li, X., and Shah, M. (2013). Simultaneous video stabilization and moving object detection in turbulence. *IEEE T-PAMI*, 35(2):450–462.

[93] Organisciak, D., Sakkos, D., Jandová, K., Ho, E. S. L., Aslam, N., and Shum, H. P. H. (Submitted in May 2019). Unifying person and vehicle re-identification. *IEEE Transactions on Circuits and Systems for Video Technology. Under review.*

[94] Pathak, D., Krähenbühl, P., Donahue, J., Darrell, T., and Efros, A. A. (2016). Context encoders: Feature learning by inpainting. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2536–2544.

[95] Patil, P. and Murala, S. (2019). Fggan: A cascaded unpaired learning for background estimation and foreground segmentation. *IEEE Winter Conference on Applications of Computer Vision (WACV)*.

[96] Perazzi, F., Pont-Tuset, J., McWilliams, B., Gool, L. V., Gross, M., and Sorkine-Hornung, A. (2016). A benchmark dataset and evaluation methodology for video object segmentation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 724–732.

[97] Pham, H., Guan, M., Zoph, B., Le, Q., and Dean, J. (2018). Efficient neural architecture search via parameters sharing. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4095–4104, Stockholmsmässan, Stockholm Sweden. PMLR.

[98] Pilet, J., Strecha, C., and Fua, P. (2008). Making background subtraction robust to sudden illumination changes. In *European Conference on Computer Vision (ECCV)*, pages 567–580.

[99] Pinheiro, P. H. O. P. and Collobert, R. (2013). Recurrent Convolutional Neural Networks for Scene Parsing. *Proceedings of The 31st International Conference on Machine Learning*, 32(June):82–90.

[100] Poynton, C. A. (1993). Gamma and its disguises: the nonlinear mappings of intensity in perception, crts, film, and video. *SMPTE*, 102:1099–1108.

[101] Qian, Y., Bi, M., Tan, T., and Yu, K. (2016). Very deep convolutional neural networks for noise robust speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(12):2263–2276.

[102] Ren, W., Liu, S., Zhang, H., Pan, J., Cao, X., and Yang, M.-H. (2016). Single image dehazing via multi-scale convolutional neural networks. In *ECCV*, pages 154–169.

[103] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241.

[104] Roy, A. and Todorovic, S. (2016). A multi-scale cnn for affordance segmentation in rgb images. In *Computer Vision – ECCV 2016*, pages 186–201, Cham. Springer International Publishing.

[105] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

[106] Sajid, H. and Cheung, S.-C. S. (2017). Universal Multimode Background Subtraction. *IEEE Transactions on Image Processing*, 26(7):3249–3260.

[107] Sakkos, D., Ho, E. S. L., and Shum, H. P. H. (2019a). Illumination-Aware Multi-Task GANs for Foreground Segmentation. *IEEE Access*, 7:10976–10986.

[108] Sakkos, D., Ho, E. S. L., and Shum, H. P. H. (Submitted in June 2019b). Synthetic data augmentation for robust background subtraction. *SKIMA. Under Review.*

[109] Sakkos, D., Liu, H., Han, J., and Shao, L. (2018). End-to-end video background subtraction with 3D convolutional neural networks. *Multimedia Tools and Applications*, 77(17):23023–23041.

[110] Scherer, D., Müller, A., and Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In *Proceedings of the 20th International Conference on Artificial Neural Networks: Part III*, ICANN'10, pages 92–101, Berlin, Heidelberg. Springer-Verlag.

[111] Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[112] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2014). Overfeat: Integrated recognition, localization and detection using convolutional networks. *ICLR*, abs/1312.6229.

[113] Shen, Y., Hu, W., Yang, M., Liu, J., Wei, B., Lucey, S., and Chou, C. (2016). Real-time and robust compressive background subtraction for embedded camera networks. *IEEE TRANSACTIONS ON MOBILE COMPUTING*, 15(2):406 – 418.

[114] Shimada, A., N. H. and Taniguchi, R. (2013). Background modeling based on bidirectional analysis. *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1979–1986.

[115] Simonyan, K. and Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations (ICRL)*, pages 1–14.

[116] Sirikuntamat, N., Satoh, S., and Chalidabhongse, T. H. (2015). Vehicle tracking in low hue contrast based on camshift and background subtraction. In *2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 58–62.

[117] Siva, P., Shafiee, M. J., Li, F., and Wong, A. (2015). Pirm: Fast background subtraction under sudden, local illumination changes via probabilistic illumination range modelling. *2015 IEEE International Conference on Image Processing (ICIP)*.

[118] Sobral, A. and Vacavant, A. (2014). A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. *Computer Vision and Image Understanding*, 122:4–21.

[119] Sondergaard, T. and Lermusiaux, P. F. J. (2013). Data assimilation with gaussian mixture models using the dynamically orthogonal field equations. part i: Theory and scheme. *Monthly Weather Review*, 141(6):1737–1760.

[120] Song, J., Yu, Q., Song, Y.-Z., Xiang, T., and Hospedales, T. M. (2017). Deep spatial-semantic attention for fine-grained sketch-based image retrieval. *2017 IEEE International Conference on Computer Vision (ICCV)*.

[121] St-Charles, P. L., Bilodeau, G. A., and Bergevin, R. (2015a). A self-adjusting approach to change detection based on background word consensus. In *Proceedings - 2015 IEEE Winter Conference on Applications of Computer Vision, WACV 2015*, pages 990–997.

[122] St-Charles, P.-L., Bilodeau, G.-A., and Bergevin, R. (2015b). SuBSENSE: a universal change detection method with local adaptive sensitivity. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 24(1):359–73.

[123] Stagliano, A., Noceti, N., Verri, A., and Odone, F. (2015). Online space-variant background modeling with sparse coding. *IEEE Transactions on Image Processing*, 24(8):2415–2428.

[124] Stauffer, C. and Grimson, W. E. L. (1999). Adaptive background mixture models for real-time tracking. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 2, pages 246–252 Vol. 2.

[125] Stefano, L. D., Tombari, F., and Mattoccia, S. (2007). Robust and accurate change detection under sudden illumination variations. *ACCV'07 Workshop on Multi-dimensional and Multi-view Image Processing, Tokyo, Nov., 2007 MM-P-02*, pages 103–109.

[126] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[127] Terrell, G. R. and Scott, D. W. (1992). Variable kernel density estimation. *The Annals of Statistics*, 20(3):1236–1265.

[128] Thangarajah, A. (2018). A foreground inference network for video surveillance using multi-view receptive field. *CoRR*, abs/1801.06593.

[129] Them, K., Kaul, M. G., Jung, C., Hofmann, M., Mummert, T., Werner, F., , and Knopp, T. (2016). Sensitivity enhancement in magnetic particle imaging by background subtraction. *IEEE TRANSACTIONS ON MEDICAL IMAGING*, 35(3).

[130] Torre, F. D. and Black, M. J. (2003). A Framework for Robust Subspace Learning. *International Journal of Computer Vision*, 54(1):117–142.

[131] Tran, D., Bourdev, L. D., Fergus, R., Torresani, L., and Paluri, M. (2014). C3D: generic features for video analysis. *CoRR*, abs/1412.0767.

[132] Tuzel, O., Porikli, F., and Meer, P. (2005). A Bayesian Approach to Background Modeling. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*, 3:58–58.

[133] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

[134] Vosters, L., Shan, C., and Gritti, T. (2010). Background subtraction under sudden illumination changes. *2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance*.

[135] Vosters, L., Shan, C., and Gritti, T. (2012). Real-time robust background subtraction under rapidly changing illumination conditions. *Image and Vision Computing*, 30(12):1004–1015.

[136] Wang, P., Chen, P., Yuan, Y., Liu, D., Huang, Z., Hou, X., and Cottrell, G. W. (2018a). Understanding convolution for semantic segmentation. *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1451–1460.

[137] Wang, R., Bunyak, F., Seetharaman, G., and Palaniappan, K. (2014a). Static and moving object detection using flux tensor with split gaussian models. In *IEEE Computer Society Conference on Computer Visxion and Pattern Recognition Workshops*, pages 420–424.

[138] Wang, X. and Gupta, A. (2016). Generative image modeling using style and structure adversarial networks. In *Computer Vision – ECCV 2016*, pages 318–335, Cham. Springer International Publishing.

[139] Wang, X., Lu, C., Jia, J., and Li, H. (2017a). l_0 regularized stationary-time estimation for crowd analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(5):981–994.

[140] Wang, Y., Jodoin, P. M., Porikli, F., Konrad, J., Benezeth, Y., and Ishwar, P. (2014b). CDnet 2014: An expanded change detection benchmark dataset. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 393–400.

[141] Wang, Y., Luo, Z., and Jodoin, P.-M. (2017b). Interactive deep learning method for segmenting moving objects. *Pattern Recognition Letters*, 96:66–75.

[142] Wang, Y., Yu, Z., and Zhu, L. (2018b). Foreground detection with deeply learned multi-scale spatial-temporal features. *Sensors*, 18(12):4269.

[143] Xie, S. and Tu, Z. (2015). Holistically-nested edge detection. In *Proceedings of IEEE International Conference on Computer Vision*.

[144] Xin, B., Kawahara, Y., Wang, Y., Hu, L., and Gao, W. (2016). Efficient generalized fused lasso and its applications to the diagnosis of alzheimers disease. *ACM Transactions on Intelligent Systems and Technology*, 7(4):1–22.

[145] Xin, B., Tian, Y., Wang, Y., and Gao, W. (2015). Background subtraction via generalized fused lasso foreground modeling. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[146] Xu, H. and Saenko, K. (2016). Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *Computer Vision – ECCV 2016*, pages 451–466, Cham. Springer International Publishing.

[147] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057, Lille, France. PMLR.

[148] Yakubovskiy, P. (2019). Segmentation models. https://github.com/qubvel/segmentation_models.

[149] Yao, C., Liu, Y. F., Jiang, B., Han, J., and Han, J. (2017). LLE score: a new filter-based unsupervised feature selection method based on nonlinear manifold embedding and its application to image recognition.

[150] Yao, X., Han, J., Cheng, G., Qian, X., and Guo, L. (2016). Semantic Annotation of High-Resolution Satellite Images via Weakly Supervised Learning. *IEEE Transactions on Geoscience and Remote Sensing*, 54(6):3660–3671.

[151] Yeo, H.-S., Lee, B.-G., and Lim, H. (2013). Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware. *Multimedia Tools and Applications*.

[152] Yi, Z., Zhang, H., Tan, P., and Gong, M. (2017). DualGAN: Unsupervised Dual Learning for Image-to-Image Translation. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2017-October, pages 2868–2876.

[153] Yong, H., Meng, D., Zuo, W., and Zhang, L. (2017). Robust online matrix factorization for dynamic background subtraction. *IEEE T-PAMI*, PP(99).

[154] Yu, F. and Koltun, V. (2016). Multi-scale context aggregation by dilated convolutions. In *ICLR*.

[155] Yu, F., Koltun, V., and Funkhouser, T. (2017). Dilated residual networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 636–644.

[156] Zeng, D. and Zhu, M. (2018a). Background subtraction using multiscale fully convolutional network. *IEEE Access*, 6:16010–16021.

[157] Zeng, D. and Zhu, M. (2018b). Multiscale fully convolutional network for foreground object detection in infrared videos. *IEEE Geoscience and Remote Sensing Letters*, 15(4):617–621.

[158] Zhang, S., Yao, H., and Liu, S. (2008a). Dynamic background modeling and subtraction using spatio-temporal local binary patterns. In *Proceedings - International Conference on Image Processing, ICIP*, pages 1556–1559.

[159] Zhang, S., Yao, H., and Liu, S. (2009). Dynamic Background Subtraction Based on Local Dependency Histogram. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(07):1397.

[160] Zhang, S., Yao, H., Liu, S., Chen, X., and Gao, W. (2008b). A covariance-based method for dynamic background subtraction. *2008 19th International Conference on Pattern Recognition*, pages 4–7.

[161] Zhao, H., Shi, J., Qi, X., Wang, X., and Jia, J. (2017a). Pyramid scene parsing network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6230–6239.

[162] Zhao, Q., Zhou, G., Zhang, L., Cichocki, A., and Amari, S.-I. (2016a). Bayesian robust tensor factorization for incomplete multiway data. *IEEE T-NNLS*, PP(99):1–1.

[163] Zhao, S., Yao, H., Gao, Y., Ji, R., Xie, W., Jiang, X., and Chua, T.-S. (2016b). Predicting personalized emotion perceptions of social images. In *Proceedings of the 2016 ACM on Multimedia Conference*, MM '16, pages 1385–1394, New York, NY, USA. ACM.

[164] Zhao, X., Chen, Y., Tang, M., and Wang, J. (2017b). Joint background reconstruction and foreground segmentation via a two-stage convolutional neural network. *CoRR*, abs/1707.07584.

[165] Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. H. S. (2015). Conditional Random Fields as Recurrent Neural Networks. *Iccv*, pages 1529–1537.

[166] Zhou, T. and Tao, D. (2011). GoDec : Randomized Low-rank & Sparse Matrix Decomposition in Noisy Case. *ICML*, page 8.

[167] Zhou, X., Yang, C., and Yu, W. (2013). Moving object detection by detecting contiguous outliers in the low-rank representation. *IEEE T-PAMI*, 35(3):597–610.

[168] Zhu, J., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251.

[169] Zhu, Q., Shao, L., Li, Q., and Xie, Y. (2013). Recursive kernel density estimation for modeling the background and segmenting moving objects. *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 1769–1772.

[170] Zivkovic, Z. (2004). Improved adaptive gaussian mixture model for background subtraction. *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*

[171] Zivkovic, Z. and Van Der Heijden, F. (2006). Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*, 27(7):773–780.

Appendix A

Chapter 1 model

```
layer {
  name: "data_1"
  type: "Python"
  top: "data_1"
  python_param {
    module: "loadCDnetAndFlip"
    layer: "Frame1_train"
  }
}

layer {
  name: "data_2"
  type: "Python"
  top: "data_2"
  python_param {
    module: "loadCDnetAndFlip"
    layer: "Frame2_train"
  }
}

layer {
  name: "data_3"
  type: "Python"
  top: "data_3"
  python_param {
    module: "loadCDnetAndFlip"
    layer: "Frame3_train"
  }
}

layer {
  name: "data_4"
  type: "Python"
  top: "data_4"
  python_param {
    module: "loadCDnetAndFlip"
    layer: "Frame4_train"
  }
}
```

```
layer {
  name: "data_5"
  type: "Python"
  top: "data_5"
  python_param {
    module: "loadCDnetAndFlip"
    layer: "Frame5_train"
  }
}

layer {
  name: "data_6"
  type: "Python"
  top: "data_6"
  python_param {
    module: "loadCDnetAndFlip"
    layer: "Frame6_train"
  }
}


layer {
  name: "data_7"
  type: "Python"
  top: "data_7"
  python_param {
    module: "loadCDnetAndFlip"
    layer: "Frame7_train"
  }
}


layer {
  name: "data_8"
  type: "Python"
  top: "data_8"
  python_param {
    module: "loadCDnetAndFlip"
    layer: "Frame8_train"
  }
}


layer {
  name: "data_9"
  type: "Python"
  top: "data_9"
  python_param {
    module: "loadCDnetAndFlip"
    layer: "Frame9_train"
  }
}

layer {
  name: "data_10"
  type: "Python"
  top: "data_10"
```

```
  top: "label"
  python_param {
    module: "loadCDnetAndFlip"
    layer: "Frame10_train"
  }
}

layer {
    name: "res1"
    type: "Reshape"
    bottom: "data_1"
    top: "res1"
    reshape_param {
      shape {
        dim: 0
        dim: 0
        dim: 1
        dim: 240
        dim: -1
      }
    }
  }
layer {
    name: "res2"
    type: "Reshape"
    bottom: "data_2"
    top: "res2"
    reshape_param {
      shape {
        dim: 0
        dim: 0
        dim: 1
        dim: 240
        dim: -1
      }
    }
}
layer {
    name: "res3"
    type: "Reshape"
    bottom: "data_3"
    top: "res3"
    reshape_param {
      shape {
        dim: 0
        dim: 0
        dim: 1
        dim: 240
        dim: -1
      }
    }
}

layer {
    name: "res4"
    type: "Reshape"
```

```
    bottom: "data_4"
    top: "res4"
    reshape_param {
      shape {
        dim: 0
        dim: 0
        dim: 1
        dim: 240
        dim: -1
      }
    }
}

layer {
    name: "res5"
    type: "Reshape"
    bottom: "data_5"
    top: "res5"
    reshape_param {
      shape {
        dim: 0
        dim: 0
        dim: 1
        dim: 240
        dim: -1
      }
    }
}

layer {
    name: "res6"
    type: "Reshape"
    bottom: "data_6"
    top: "res6"
    reshape_param {
      shape {
        dim: 0
        dim: 0
        dim: 1
        dim: 240
        dim: -1
      }
    }
}
layer {
    name: "res7"
    type: "Reshape"
    bottom: "data_7"
    top: "res7"
    reshape_param {
      shape {
        dim: 0
        dim: 0
        dim: 1
        dim: 240
        dim: -1
```

```
      }
    }
}
layer {
    name: "res8"
    type: "Reshape"
    bottom: "data_8"
    top: "res8"
    reshape_param {
      shape {
        dim: 0
        dim: 0
        dim: 1
        dim: 240
        dim: -1
      }
    }
}
layer {
    name: "res9"
    type: "Reshape"
    bottom: "data_9"
    top: "res9"
    reshape_param {
      shape {
        dim: 0
        dim: 0
        dim: 1
        dim: 240
        dim: -1
      }
    }
}

layer {
    name: "res10"
    type: "Reshape"
    bottom: "data_10"
    top: "res10"
    reshape_param {
      shape {
        dim: 0
        dim: 0
        dim: 1
        dim: 240
        dim: -1
      }
    }
}

layer {
  name: "merge1"
  type: "Concat"
  bottom: "res1"
  bottom: "res2"
  bottom: "res3"
```

```
  bottom: "res4"
  top: "merge1"
  concat_param {
    axis:2
  }
}

layer {
  name: "merge2"
  type: "Concat"
  bottom: "res3"
  bottom: "res4"
  bottom: "res5"
  bottom: "res6"
  top: "merge2"
  concat_param {
    axis:2
  }
}


layer {
  name: "merge3"
  type: "Concat"
  bottom: "res5"
  bottom: "res6"
  bottom: "res7"
  bottom: "res8"
  top: "merge3"
  concat_param {
    axis:2
  }
}

layer {
  name: "merge4"
  type: "Concat"
  bottom: "res7"
  bottom: "res8"
  bottom: "res9"
  bottom: "res10"
  top: "merge4"
  concat_param {
    axis:2
  }
}
layer {
  name: "conv1_1"
  type: "Convolution3D"
  bottom: "merge1"
  top: "conv1_1"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
```

```
    lr_mult: 2
    decay_mult: 0
  }
  convolution3d_param {
    num_output: 64
    pad: 1
    kernel_size: 3
    kernel_depth: 4
    stride: 1
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
    temporal_pad: 1
  }
}

layer {
  name: "relu1_1"
  type: "ReLU"
  bottom: "conv1_1"
  top: "conv1_1"
}

layer {
  name: "pool1_1"
  type: "Pooling3D"
  bottom: "conv1_1"
  top: "pool1_1"
  pooling3d_param {
    pool: MAX
    kernel_size: 2
    kernel_depth: 1
    stride: 2
    temporal_stride: 1
  }
}
layer {
  name: "conv1_2"
  type: "Convolution3D"
  bottom: "merge2"
  top: "conv1_2"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution3d_param {
    num_output: 64
```

```
    pad: 1
    temporal_pad: 1
    kernel_size: 3
    kernel_depth: 4
    stride: 1
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
layer {
  name: "relu1_2"
  type: "ReLU"
  bottom: "conv1_2"
  top: "conv1_2"
}

layer {
  name: "pool1_2"
  type: "Pooling3D"
  bottom: "conv1_2"
  top: "pool1_2"
  pooling3d_param {
    pool: MAX
    kernel_size: 2
    kernel_depth: 1
    stride: 2
    temporal_stride: 1
  }
}

layer {
  name: "conv1_3"
  type: "Convolution3D"
  bottom: "merge3"
  top: "conv1_3"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution3d_param {
    num_output: 64
    pad: 1
    temporal_pad: 1
    kernel_size: 3
    kernel_depth: 4
    stride: 1
```

```
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
layer {
  name: "relu1_3"
  type: "ReLU"
  bottom: "conv1_3"
  top: "conv1_3"
}

layer {
  name: "pool1_3"
  type: "Pooling3D"
  bottom: "conv1_3"
  top: "pool1_3"
  pooling3d_param {
    pool: MAX
    kernel_size: 2
    kernel_depth: 1
    stride: 2
    temporal_stride: 1
  }
}

layer {
  name: "conv1_4"
  type: "Convolution3D"
  bottom: "merge4"
  top: "conv1_4"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution3d_param {
    num_output: 64
    pad: 1
    temporal_pad: 1
    kernel_size: 3
    kernel_depth: 4
    stride: 1
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
```

```
      type: "constant"
      value: 0
    }
  }
}
layer {
  name: "relu1_4"
  type: "ReLU"
  bottom: "conv1_4"
  top: "conv1_4"
}

layer {
  name: "pool1_4"
  type: "Pooling3D"
  bottom: "conv1_4"
  top: "pool1_4"
  pooling3d_param {
    pool: MAX
    kernel_size: 2
    kernel_depth: 1
    stride: 2
    temporal_stride: 1
  }
}

layer {
  name: "merge2_1"
  type: "Concat"
  bottom: "pool1_1"
  bottom: "pool1_2"
  top: "merge2_1"
  concat_param {
    axis:2
  }
}
layer {
  name: "merge2_2"
  type: "Concat"
  bottom: "pool1_3"
  bottom: "pool1_4"
  top: "merge2_2"
  concat_param {
    axis:2
  }
}

layer {
  name: "conv2_1"
  type: "Convolution3D"
  bottom: "merge2_1"
  top: "conv2_1"
  param {
    lr_mult: 1
    decay_mult: 1
  }
```

```
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution3d_param {
    num_output: 128
    pad: 1
    temporal_pad: 1
    kernel_size: 3
    kernel_depth: 2
    stride: 1
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
layer {
  name: "relu2_1"
  type: "ReLU"
  bottom: "conv2_1"
  top: "conv2_1"
}

layer {
  name: "pool2_1"
  type: "Pooling3D"
  bottom: "conv2_1"
  top: "pool2_1"
  pooling3d_param {
    pool: MAX
    kernel_size: 2
    kernel_depth: 1
    stride: 2
    temporal_stride: 1
  }
}

layer {
  name: "conv2_2"
  type: "Convolution3D"
  bottom: "merge2_2"
  top: "conv2_2"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
```

```
  }
  convolution3d_param {
    num_output: 128
    pad: 1
    temporal_pad: 1
    kernel_size: 3
    kernel_depth: 2
    stride: 1
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
layer {
  name: "relu2_2"
  type: "ReLU"
  bottom: "conv2_2"
  top: "conv2_2"
}

layer {
  name: "pool2_2"
  type: "Pooling3D"
  bottom: "conv2_2"
  top: "pool2_2"
  pooling3d_param {
    pool: MAX
    kernel_size: 2
    kernel_depth: 1
    stride: 2
    temporal_stride: 1
  }
}

layer {
  name: "merge3_1"
  type: "Concat"
  bottom: "pool2_1"
  bottom: "pool2_2"
  top: "merge3_1"
  concat_param {
    axis:2
  }
}

layer {
  name: "conv3"
  type: "Convolution3D"
  bottom: "merge3_1"
```

```
  top: "conv3"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution3d_param {
    num_output: 256
    pad: 1
    temporal_pad: 1
    kernel_size: 3
    kernel_depth: 16
    stride: 1
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
layer {
  name: "relu3"
  type: "ReLU"
  bottom: "conv3"
  top: "conv3"
}
layer {
    name: "res"
    type: "Reshape"
    bottom: "conv3"
    top: "res"
    reshape_param {
      shape {
        dim: 0
        dim: 0
        dim: 60
        dim: -1
      }
    }
}

layer {
  name: "pool3"
  type: "Pooling"
  bottom: "res"
  top: "pool3"
  pooling_param {
    pool: MAX
    kernel_size: 2
    stride: 2
```

```
  }
}
layer { bottom: 'pool3' top: 'conv4_1' name: 'conv4_1' type: "Convolution"
  param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0}
  convolution_param {num_output: 512 pad: 1 kernel_size: 3 weight_filler {
        type: "xavier"
      }
      bias_filler {
        type: "constant"
}} }
layer { bottom: 'conv4_1' top: 'conv4_1' name: 'relu4_1' type: "ReLU" }
layer { bottom: 'conv4_1' top: 'conv4_2' name: 'conv4_2' type: "Convolution"
  param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0}
  convolution_param {num_output: 512 pad: 1 kernel_size: 3
weight_filler {
        type: "xavier"
      }
      bias_filler {
        type: "constant"
}} }
layer { bottom: 'conv4_2' top: 'conv4_2' name: 'relu4_2' type: "ReLU" }
layer { bottom: 'conv4_2' top: 'conv4_3' name: 'conv4_3' type: "Convolution"
  param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0}
  convolution_param {num_output: 512 pad: 1 kernel_size: 3
weight_filler {
        type: "xavier"
      }
      bias_filler {
        type: "constant"
}} }
layer { bottom: 'conv4_3' top: 'conv4_3' name: 'relu4_3' type: "ReLU" }

layer { bottom: 'conv4_3' top: 'pool4' name: 'pool4' type: "Pooling"
  pooling_param { pool: MAX kernel_size: 2 stride: 2 } }


layer { bottom: 'pool4' top: 'conv5_1' name: 'conv5_1' type: "Convolution"
  param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0}
  convolution_param {num_output: 512 pad: 1 kernel_size: 3
weight_filler {
        type: "xavier"
      }
      bias_filler {
        type: "constant"
}} }
layer { bottom: 'conv5_1' top: 'conv5_1' name: 'relu5_1' type: "ReLU" }
layer { bottom: 'conv5_1' top: 'conv5_2' name: 'conv5_2' type: "Convolution"
  param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0}
  convolution_param {num_output: 512 pad: 1 kernel_size: 3
weight_filler {
        type: "xavier"
      }
      bias_filler {
        type: "constant"
}} }
layer { bottom: 'conv5_2' top: 'conv5_2' name: 'relu5_2' type: "ReLU" }
```

```
layer { bottom: 'conv5_2' top: 'conv5_3' name: 'conv5_3' type: "Convolution"
  param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0}
  convolution_param {num_output: 512 pad: 1 kernel_size: 3
weight_filler {
        type: "xavier"
      }
      bias_filler {
        type: "constant"
}} }
layer { bottom: 'conv5_3' top: 'conv5_3' name: 'relu5_3' type: "ReLU" }

layer {
  name: "merge_c_2"
  type: "Concat"
  bottom: "conv2_1"
  bottom: "conv2_2"
  top: "merge_c_2"
  concat_param {
    axis:2
  }
}


layer {
  name: "conv2_2_16_"
  type: "Convolution3D"
  bottom: "merge_c_2"
  top: "conv2_2_16"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution3d_param {
    num_output: 1
    pad: 1
    temporal_pad: 1
    kernel_size: 3
    kernel_depth: 16
    stride: 1
    weight_filler {
      type: "gaussian"
      std: 0.001
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}

layer {
```

```
    name: "res216"
    type: "Reshape"
    bottom: "conv2_2_16"
    top: "res216"
    reshape_param {
      shape {
        dim: 0
        dim: 0
        dim: 120
        dim: -1
      }
    }
}

## Prep 3
layer { bottom: 'res' top: 'conv3_3_16' name: 'conv3_3_16_'
type: "Convolution" param { lr_mult: 1 decay_mult: 1 }
param { lr_mult: 2 decay_mult: 0} convolution_param {num_output: 1 pad: 1
kernel_size: 3 weight_filler{ type: "gaussian" std: 0.001}} }


## Prep 4
layer { bottom: 'conv4_3' top: 'conv4_3_16' name: 'conv4_3_16_'
type: "Convolution" param { lr_mult: 1 decay_mult: 1 }
param { lr_mult: 2 decay_mult: 0} convolution_param {num_output: 1 pad: 1
kernel_size: 3 weight_filler{ type: "gaussian" std: 0.001}} }


## Prep 5
layer { bottom: 'conv5_3' top: 'conv5_3_16' name: 'conv5_3_16_'
type: "Convolution" param { lr_mult: 1 decay_mult: 1 }
  param { lr_mult: 2 decay_mult: 0} convolution_param {num_output: 1 pad: 1
  kernel_size: 3 weight_filler{ type: "gaussian" std: 0.001}} }


### Multiple conv 2 ###
layer { type: "Deconvolution" name: 'upsample2' bottom: 'res216'
top: 'side-multi2-up'
  param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0}
  convolution_param { kernel_size: 4 stride: 2 num_output: 1 pad: 1 } }
layer { type: "Crop" name: 'crop' bottom: 'side-multi2-up'
bottom: 'data_10' top: 'upside-multi2' }

### Multiple conv 3 ###
layer { type: "Deconvolution" name: 'upsample4' bottom: 'conv3_3_16'
top: 'side-multi3-up'
  param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0}
  convolution_param { kernel_size: 8 stride: 4 num_output: 1  } }
layer { type: "Crop" name: 'crop'  bottom: 'side-multi3-up'
bottom: 'data_10' top: 'upside-multi3'   crop_param {
    axis: 2
    offset: 2
  }}

### Multiple conv 4 ###
layer { type: "Deconvolution" name: 'upsample8' bottom: 'conv4_3_16'
top: 'side-multi4-up'
```

```
  param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0}
  convolution_param { kernel_size: 16 stride: 8 num_output: 1  } }
layer { type: "Crop" name: 'crop' bottom: 'side-multi4-up'
bottom: 'data_10' top: 'upside-multi4'   crop_param {
    axis: 2
    offset: 4
  }}

### Multiple conv 5 ###
layer { type: "Deconvolution" name: 'upsample16' bottom: 'conv5_3_16'
top: 'side-multi5-up'
  param { lr_mult: 1 decay_mult: 1 } param { lr_mult: 2 decay_mult: 0}
  convolution_param { kernel_size: 32 stride: 16 num_output: 1  } }
layer { type: "Crop" name: 'crop' bottom: 'side-multi5-up'
bottom: 'data_10' top: 'upside-multi5'   crop_param {
    axis: 2
    offset: 8
    offset: 8
  }}

### Concat and multiscale weight layer ###
layer { name: "concat" bottom: "upside-multi2"
bottom: "upside-multi3" bottom: "upside-multi4" bottom: "upside-multi5"
top: "concat-upscore" type: "Concat"
  concat_param { concat_dim: 1} }
layer { name: 'new-score-weighting_' type: "Convolution"
bottom: 'concat-upscore' top: 'upscore-fuse'
  param { lr_mult: 0.01 decay_mult: 0 } param { lr_mult: 0.02 decay_mult: 0}
  convolution_param {num_output: 1 kernel_size: 1
  weight_filler {type: "constant" value: 0.0} } }
layer { type: "SigmoidCrossEntropyLoss" bottom: "upscore-fuse"
bottom: "label" top:"fuse_loss" loss_param {ignore_label: 2}
loss_weight: 1 }
```

Appendix B

Chapter 2 model

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, None, None, 3)     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, None, None, 64)    1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, None, None, 64)    36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, None, None, 64)    0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, None, None, 128)   73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, None, None, 128)   147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, None, None, 128)   0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, None, None, 256)   295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, None, None, 256)   590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, None, None, 256)   590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, None, None, 256)   0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, None, None, 512)   1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, None, None, 512)   2359808
```

```
_________________________________________________________________
block4_conv3 (Conv2D)                (None, None, None, 512)
2359808         block4_conv2[0][0]
_________________________________________________________________
block4_pool (MaxPooling2D)           (None, None, None, 512)
0               block4_conv3[0][0]
_________________________________________________________________
block5_conv1 (Conv2D)                (None, None, None, 512)
2359808         block4_pool[0][0]
_________________________________________________________________
block5_conv2 (Conv2D)                (None, None, None, 512)
2359808         block5_conv1[0][0]
_________________________________________________________________
block5_conv3 (Conv2D)                (None, None, None, 512)
2359808         block5_conv2[0][0]
_________________________________________________________________
block5_pool (MaxPooling2D)           (None, None, None, 512)
0               block5_conv3[0][0]
_________________________________________________________________
decoder_stage0_upsample (UpSampling2D)  (None, None, None, 512)
0               block5_pool[0][0]
_________________________________________________________________
concatenate_1 (Concatenate)          (None, None, None, 1024)
0               decoder_stage0_upsample[0][0]

_________________________________________________________________
decoder_stage0_conv1 (Conv2D)        (None, None, None, 256)
2359296         concatenate_1[0][0]
_________________________________________________________________
decoder_stage0_bn1 (BatchNormalization)  (None, None, None, 256)
1024            decoder_stage0_conv1[0][0]
_________________________________________________________________
decoder_stage0_relu1 (Activation)    (None, None, None, 256)
0               decoder_stage0_bn1[0][0]
_________________________________________________________________
decoder_stage0_conv2 (Conv2D)        (None, None, None, 256)
589824          decoder_stage0_relu1[0][0]
_________________________________________________________________
decoder_stage0_bn2 (BatchNormalization)  (None, None, None, 256)
1024            decoder_stage0_conv2[0][0]
_________________________________________________________________
decoder_stage0_relu2 (Activation)    (None, None, None, 256)
0               decoder_stage0_bn2[0][0]
_________________________________________________________________
decoder_stage1_upsample (UpSampling2D)  (None, None, None, 256)
0               decoder_stage0_relu2[0][0]
_________________________________________________________________
concatenate_2 (Concatenate)          (None, None, None, 768)
0               decoder_stage1_upsample[0][0]

_________________________________________________________________
decoder_stage1_conv1 (Conv2D)        (None, None, None, 128)
884736          concatenate_2[0][0]
_________________________________________________________________
decoder_stage1_bn1 (BatchNormalization)  (None, None, None, 128)
512             decoder_stage1_conv1[0][0]
_________________________________________________________________
```

```
decoder_stage1_relu1 (Activation)               (None, None, None, 128)
0                decoder_stage1_bn1[0][0]
 _____________________________________________________________________
decoder_stage1_conv2 (Conv2D)                   (None, None, None, 128)
147456           decoder_stage1_relu1[0][0]
 _____________________________________________________________________
decoder_stage1_bn2 (BatchNormalization)         (None, None, None, 128)
512              decoder_stage1_conv2[0][0]
 _____________________________________________________________________
decoder_stage1_relu2 (Activation)               (None, None, None, 128)
0                decoder_stage1_bn2[0][0]
 _____________________________________________________________________
decoder_stage2_upsample (UpSampling2D)          (None, None, None, 128)
0                decoder_stage1_relu2[0][0]
 _____________________________________________________________________
concatenate_3 (Concatenate)                     (None, None, None, 384)
0                decoder_stage2_upsample[0][0]

 _____________________________________________________________________
decoder_stage2_conv1 (Conv2D)                   (None, None, None, 64)
221184           concatenate_3[0][0]
 _____________________________________________________________________
decoder_stage2_bn1 (BatchNormalization)         (None, None, None, 64)
256              decoder_stage2_conv1[0][0]
 _____________________________________________________________________
decoder_stage2_relu1 (Activation)               (None, None, None, 64)
0                decoder_stage2_bn1[0][0]
 _____________________________________________________________________
decoder_stage2_conv2 (Conv2D)                   (None, None, None, 64)
36864            decoder_stage2_relu1[0][0]
 _____________________________________________________________________
decoder_stage2_bn2 (BatchNormalization)         (None, None, None, 64)
256              decoder_stage2_conv2[0][0]
 _____________________________________________________________________
decoder_stage2_relu2 (Activation)               (None, None, None, 64)
0                decoder_stage2_bn2[0][0]
 _____________________________________________________________________
decoder_stage3_upsample (UpSampling2D)          (None, None, None, 64)
0                decoder_stage2_relu2[0][0]
 _____________________________________________________________________
concatenate_4 (Concatenate)                     (None, None, None, 192)
0                decoder_stage3_upsample[0][0]

 _____________________________________________________________________
decoder_stage3_conv1 (Conv2D)                   (None, None, None, 32)
55296            concatenate_4[0][0]
 _____________________________________________________________________
decoder_stage3_bn1 (BatchNormalization)         (None, None, None, 32)
128              decoder_stage3_conv1[0][0]
 _____________________________________________________________________
decoder_stage3_relu1 (Activation)               (None, None, None, 32)
0                decoder_stage3_bn1[0][0]
 _____________________________________________________________________
decoder_stage3_conv2 (Conv2D)                   (None, None, None, 32)
9216             decoder_stage3_relu1[0][0]
 _____________________________________________________________________
decoder_stage3_bn2 (BatchNormalization)         (None, None, None, 32)
128              decoder_stage3_conv2[0][0]
 _____________________________________________________________________
```

```
decoder_stage3_relu2 (Activation)               (None, None, None, 32)
0                decoder_stage3_bn2[0][0]
__________________________________________________________________________
decoder_stage4_upsample (UpSampling2D)          (None, None, None, 32)
0                decoder_stage3_relu2[0][0]
__________________________________________________________________________
decoder_stage4_conv1 (Conv2D)                   (None, None, None, 16)
4608             decoder_stage4_upsample[0][0]
__________________________________________________________________________
decoder_stage4_bn1 (BatchNormalization)         (None, None, None, 16)
64               decoder_stage4_conv1[0][0]
__________________________________________________________________________
decoder_stage4_relu1 (Activation)               (None, None, None, 16)
0                decoder_stage4_bn1[0][0]
__________________________________________________________________________
decoder_stage4_conv2 (Conv2D)                   (None, None, None, 16)
2304             decoder_stage4_relu1[0][0]
__________________________________________________________________________
decoder_stage4_bn2 (BatchNormalization)         (None, None, None, 16)
64               decoder_stage4_conv2[0][0]
__________________________________________________________________________
decoder_stage4_relu2 (Activation)               (None, None, None, 16)
0                decoder_stage4_bn2[0][0]
__________________________________________________________________________
final_conv (Conv2D)                             (None, None, None, 1)
145              decoder_stage4_relu2[0][0]
__________________________________________________________________________
sigmoid (Activation)                            (None, None, None, 1)
0                final_conv[0][0]
==========================================================================
Total params: 19,029,585
Trainable params: 4,312,913
Non-trainable params: 14,716,672
__________________________________________________________________________
```

Appendix C

Chapter 3 model

C.1 Segmentation network

```
filters = 64

def segmentor(self, b5, b6, b7, d5, d6, d7, reuse):
    with tf.variable_scope(tf.get_variable_scope()) as scope:
        if reuse:
            scope.reuse_variables()
        else:
            assert scope.reuse == False
        ww,h = [self.image_height, self.image_width]
        w2, w4, w8 = int(ww / 2), int(ww / 4), int(ww / 8)
        h2, h4, h8= int(h / 2), int(h / 4), int(h / 8)

        bd7 = tf.subtract(b7,d7)
        bd7 = tf.nn.tanh(bd7)
        bd6 = tf.subtract(b6,d6)

        bd6 = tf.nn.tanh(bd6)
        bd5 = tf.subtract(b5,d5)
        bd5 = tf.nn.tanh(bd5)

        bg1 = batch_norm(conv2d(lrelu(bd7), filters*2))
        att1a = tf.reduce_mean(bg1, reduction_indices=3, keepdims=True)
        att1a = batch_norm(conv2d(lrelu(att1a), 1,k_h=3, k_w=3))
        att1a = tf.image.resize_bilinear(att1a, [w4, h4])
        att1a = batch_norm(conv2d(lrelu(att1a),
                1, k_h=1, k_w=1, d_h=1, d_w=1))

        att1b = tf.layers.average_pooling2d(bg1, [w4, h4], [w4, h4])
        att1b = batch_norm(conv2d(lrelu(att1b),
                filters/8,k_h=3, k_w=3, d_h=1, d_w=1))
        att1b = batch_norm(conv2d(lrelu(att1b),
                filters*2,k_h=3, k_w=3, d_h=1, d_w=1))

        att1 = tf.multiply(att1a, att1b)
        att1 = batch_norm(conv2d(lrelu(att1),
                filters*2,k_h=1, k_w=1, d_h=1, d_w=1))
        att1 = tf.sigmoid(att1)
        bg1 = tf.multiply(att1, bg1)
```

```
bg2 = batch_norm(conv2d(lrelu(tf.concat([bg1,bd6], 3)),
        filters*4))
att2a = tf.reduce_mean(bg2, reduction_indices=3, keepdims=True)
att2a = batch_norm(conv2d(lrelu(att2a), 1,k_h=3, k_w=3))
att2a = tf.image.resize_bilinear(att2a, [w8, h8])
att2a = batch_norm(conv2d(lrelu(att2a),
        1, k_h=1, k_w=1, d_h=1, d_w=1))

att2b = tf.layers.average_pooling2d(bg2, [w8, h8], [w8, h8])
att2b = batch_norm(conv2d(lrelu(att2b),
        filters/4,k_h=3, k_w=3, d_h=1, d_w=1))
att2b = batch_norm(conv2d(lrelu(att2b),
        filters*4,k_h=3, k_w=3, d_h=1, d_w=1))

att2 = tf.multiply(att2a, att2b)
att2 = batch_norm(conv2d(lrelu(att2),
        filters*4,k_h=1, k_w=1, d_h=1, d_w=1))
att2 = tf.sigmoid(att2)
bg2 = tf.multiply(att2, bg2)

bg3 = batch_norm(conv2d(lrelu(tf.concat([bg2,bd5], 3)),
        filters*8, k_h=3, k_w=3,d_h=1, d_w=1))
att3a = tf.reduce_mean(bg3, reduction_indices=3, keepdims=True)
att3a = batch_norm(conv2d(lrelu(att3a), 1,k_h=3, k_w=3))
att3a = tf.image.resize_bilinear(att3a, [w8, h8])
att3a = batch_norm(conv2d(lrelu(att3a),
        1, k_h=1, k_w=1, d_h=1, d_w=1))

att3b = tf.layers.average_pooling2d(bg3, [w8, h8], [w8, h8])
att3b = batch_norm(conv2d(lrelu(att3b),
        filters/2,k_h=3, k_w=3, d_h=1, d_w=1))
att3b = batch_norm(conv2d(lrelu(att3b),
        filters*8,k_h=3, k_w=3, d_h=1, d_w=1))

att3 = tf.multiply(att3a, att3b)
att3 = batch_norm(conv2d(lrelu(att3),
       filters*8,k_h=1, k_w=1, d_h=1, d_w=1))
att3 = tf.sigmoid(att3)
bg3 = tf.multiply(att3, bg3)

bg4 = batch_norm(conv2d(lrelu(bg3), filters,
        k_h=1, k_w=1,d_h=1, d_w=1))
bg4a, w, d = deconv2d(tf.nn.relu(bg4),
        [batch_size, w4, h4, filters], with_w=True)
bg4 = batch_norm(bg4a)
bg4 = batch_norm(conv2d(lrelu(bg4),
        filters * 8, k_h=1, k_w=1,d_h=1, d_w=1))

bg5 = batch_norm(conv2d(lrelu(bg4),
     filters, k_h=1, k_w=1,d_h=1, d_w=1))
bg5a, w, d = deconv2d(tf.nn.relu(bg5),
        [batch_size, w2, h2, filters], with_w=True)
bg5 = batch_norm(bg5a)
bg5 = batch_norm(conv2d(lrelu(bg5),
        filters * 8, k_h=1, k_w=1,d_h=1, d_w=1))
```

```
        bg6 = batch_norm(conv2d(lrelu(bg5),
                filters, k_h=1, k_w=1,d_h=1, d_w=1))
        bg6a, w, d = deconv2d(tf.nn.relu(bg6),
                [batch_size, ww, h, filters], with_w=True)
        bg6 = batch_norm(bg6a)
        bg6 = batch_norm(conv2d(lrelu(bg6),
                filters * 8, k_h=1, k_w=1,d_h=1, d_w=1))

        bg7 = batch_norm(conv2d(lrelu(bg6),
                filters, k_h=3, k_w=3, d_h=1, d_w=1))

        bg8 = batch_norm(conv2d(lrelu(bg7),
                1, k_h=1, k_w=1, d_h=1, d_w=1))

        bg8 = tf.nn.sigmoid(bg8)

    return bg8
```

C.2 Discriminator

```
df_dim = 64

def discriminator(self, image,  y=None, prefix='A_d_', reuse=False):

    with tf.variable_scope(tf.get_variable_scope()) as scope:
        if reuse:
            scope.reuse_variables()
        else:
            assert scope.reuse == False

        h0 = lrelu(conv2d(image, df_dim))

        h1 = lrelu(batch_norm(conv2d(h0, df_dim * 2)))

        h2 = lrelu(batch_norm(conv2d(h1, df_dim * 4)))

        h3 = lrelu(batch_norm(conv2d(h2, df_dim * 8, d_h=1, d_w=1)))

        h4 = conv2d(h3, 1, d_h=1, d_w=1)

        return h4
```

C.3 VGG and Generator

```
def vgg(self, imgs, prefix, reuse_vgg=True, reuse_up=True):
    vgg = slim.nets.vgg
    with slim.arg_scope(vgg.vgg_arg_scope()) as scope:
        logits, endpoints = vgg.vgg_16(imgs,
        is_training=self.phase == 'train',
        dropout_keep_prob=0.5, reuse=reuse_vgg)

        ww,h = [height, width]
        w2, w4, w8 = int(round(ww/2)), int(round(ww/4)), int(round(ww/8))
        h2, h4, h8 = int(round(h/2)), int(round(h/4)), int(round(h/8))
```

```
vgg512 = endpoints['vgg_16/conv4/conv4_3']
vgg256 = endpoints['vgg_16/conv3/conv3_3']
vgg128 = endpoints['vgg_16/conv2/conv2_2']
vgg64 = endpoints['vgg_16/conv1/conv1_2']

with tf.variable_scope(tf.get_variable_scope()) as scope:
    if reuse_up:
        scope.reuse_variables()
    else:
        assert scope.reuse == False
    self.d4, self.d4_w, self.d4_b = deconv2d(tf.nn.relu(vgg512),
        [batch_size, w8, h8, filters*8], d_h=1, d_w=1,with_w=True)
    d4 = batch_norm(self.d4)

    self.d5, self.d5_w, self.d5_b = deconv2d(tf.nn.relu(d4),
        [batch_size, w4, h4, filters*4], with_w=True)
    d5 = batch_norm(self.d5)
    d5 = tf.concat([d5, vgg256],3)

    self.d6, self.d6_w, self.d6_b = deconv2d(tf.nn.relu(d5),
        [batch_size, w2, h2, filters*2], with_w=True)
    d6 = batch_norm(self.d6)
    d6 = tf.concat([d6, vgg128],3)

    self.d7, self.d7_w, self.d7_b = deconv2d(tf.nn.relu(d6),
        [batch_size, ww, h, filters], with_w=True)
    d7 = batch_norm(self.d7)
    d7 = tf.concat([d7, vgg64],3)

    d8, d8_w, d8_b = deconv2d(tf.nn.relu(d7),
            [batch_size, height, width, 3],
            d_h=1, d_w=1,k_h=1, k_w=1, with_w=True)
    d8 = tf.nn.tanh(d8)

return d4,d5,d6,d8
```